Parametric Sound Texture Generator

Diplomarbeit

an der

Universität für Musik und darstellende Kunst, Graz
Technische Universität Graz

vorgelegt von

Gerda Strobl

Institut für Elektronische Musik und Akustik (IEM),
Universität für Musik und darstellende Kunst
A-8010 Graz

8. Jänner 2007

© Copyright 2007, Gerda Strobl

Diese Arbeit ist in englischer Sprache verfasst.

Begutachter: Prof. Dr. Gerhard Eckel
Mitbetreuender Professor: Prof. Dr. Davide Rocchesso (Università di Verona)

Abstract

Sound texture modeling is a widely used concept in computer music. Although, a concrete definition of sound texture is elusive, with this thesis I try to determine the scope of the different fields of acoustic texture. After the report on the current state of different sound texture generation methods I will outline common problems of the sound texture examples.

From the presented literature two existing algorithms, namely *audio texture* and *natural grains*, which are based on a similar granular analysis /resynthesis approach will be further investigated. Both algorithms can be used for creating a parametric sound texture generator that allows creating sounds out of a simple parameter control structure. Starting from a short input texture, different, new sound textures of variable length are produced. These new textures consist of consecutively connected segments that are similar to the input texture.

A special characteristic of the selected algorithms is their property of segmenting the original signal into perceptually meaningful units. Hence, sound textures which are rather associated as soundscapes that consist of different streams are particularly examined. The analysis and the resynthesis of the algorithms is effectively improved by means of parametric modifications so that soundscapes with distinct events are broken up only at points that make sense to our auditory perception.

The implementation goal of this thesis is a real-time Pure Data interface for demonstration purposes allowing the user to manually adjust parameters until the produced sound texture sounds plausible and realistic with respect to a particular use of the texture. Ideally, the sound textures are not recognized as a resynthesized version, new sounds are considered as being natural, segments are concatenated in a natural flow and no disturbing repetitive patterns or artefacts can be heard.

Kurzfassung

Die Modellierung von Klangtextur ist ein bekanntes Konzept in der Computermusik. Den Terminus Klangtextur *(sound texture)* in einer klaren Definition zu fassen ist schwierig, dennoch versuche ich in dieser Arbeit die unterschiedlichen Gebiete von akustischer Textur klar herauszuarbeiten. Nach einem Überblick über den aktuellen Stand der Forschung von unterschiedlichen Methoden der Klangtexturerzeugung, skizziere ich allgemeine Probleme von algorithmisch generierten Klangtexturen.

Zwei Algorithmen, *audio texture und natural grains*, die jeweils auf einem granularen Analyse/Resynthese-Verfahren aufbauen, werden genauer untersucht. Beide Algorithmen können als Grundlage für einen parametrischen Texturgenerator herangezogen werden. Dabei sollen aus einem kurzen Originalsignal neue Klangtexturen erzeugt werden, die aus Segmenten des Originalsignals bestehen, jedoch in neuer Ordnung aneinander gereiht werden. Die neuen Texturen sind theoretisch von unbegrenzter Dauer und dem ursprünglichen Signal ähnlich.

Eine besondere Eigenschaft der ausgewählten Algorithmen ist die Fähigkeit das Originalsignal in Segmente zu unterteilen, die mit der wahrnehmungsspezifischen Ereignisauswahl unseres Gehörs übereinstimmen. Es werden somit Klangtexturen untersucht, die sich aus mehreren Schichten von Ereignissen zusammensetzen und eher als Umgebungs- bzw. Hintergrundgeräusche bezeichnet werden. Um eine klangliche Verbesserung der Texturen zu erreichen, werden die algorithmischen Parameter solange modifiziert, bis die Segmentierung der Signale nur an Stellen erfolgt, die von der auditiven Wahrnehmung als sinnvoll erachtet werden.

Das Ziel dieser Arbeit ist eine Echtzeit-Implementierung in Pure Data. Ein graphisches Interface für Demonstrationszwecke wird präsentiert, das den BenutzerInnen ermöglicht Klangtexturen zu erzeugen und gleichzeitig Analyseparameter solange zu modifizieren bis die Texturen plausibel und natürlich klingen. Idealerweise werden die neu erzeugten Klangtexturen nicht als resynthetisierte Versionen erkannt. Die Segmente werden so aneinander gereiht, dass es zu einem klanglichen Fluss kommt und keine akustischen Artefakte entstehen.

Contents

List of Figures

Acknowledgements

Many thanks to my advisor, Gerhard Eckel, for his marvellous ideas for my work whenever he listened to my sound examples and for supporting me with so many scientific hints.

I especially wish to thank my advisor in Verona, Davide Rocchesso, for his great idea to work on sound textures, for his ability to motivate me with lots of different ideas and papers and his immediate attention to my questions. I am indebted to my wonderful colleagues Amalia de Götzen, Antonio de Sena, Carlo Drioli, Pietro Polotti and Federico Fontana at the VIPS group from the University of Verona who have provided invaluable help and feedback during the course of my work in their group.

I also like to thank Georg Holzmann for his unconventional help and also Deniz Peters, Johannes Zmölnig, Alois Sontacchi and Franz Zotter for providing lots of valuable information.

Special mention goes to Hans Pfeiffer, Birgit Gasteiger, Barbara Semmler, Stefanie Greimel, Nils Peters, Fabien Gouyon, Giuditta Franco, Mnacho Echenim, Fausto Spoto, Christina Leitner, Christoph Gratl, Roman Sereinig, Katrin Nesper, Brigitte Bergner, Nicola Bernadini, Nils Peters, Rabtaldirndln and Rosi Degen for correcting the English of this thesis.

Last but not least, without the support of my whole family, who always encouraged my thirst for knowledge, this thesis would not have been possible.

Gerda Strobl
Graz, January 2006

Credits

For the creation of this thesis I intended to use mainly open-source software. The thesis was written with the LaTex environment kile[1] and plots were made with dia[2]. The spectrograms were created with praat[3]. Furthermore, the sound files for the data base were edited with audacity[4] and the sound texture generator was built with the real-time programming language Pure Data[5].

I would like to thank the following individuals and organisations for permission to use their material: This thesis was written using Keith Andrews' LaTex skeleton thesis [Andrews, 2006]. The following figures are used subject to the public domain from Wikimedia Commons:

- Figure 1.3 extracted from:
  `http://de.wikipedia.org/wiki/Bild:Utah_teapot.png` (accessed Nov. 15. 2006)
- Figure 3.12 extracted from:
  `http://en.wikipedia.org/wiki/Image:Wavelets_-_Filter_Bank.png` (accessed Nov. 15. 2006)
- Figure 3.13 extracted from:
  `http://en.wikipedia.org/wiki/Image:Wavelets_-_DWT_Freq.png` (accessed Nov. 15. 2006)
- Figure A.1 extracted from:
  `http://upload.wikimedia.org/wikipedia/en/2/20/Mel-Hz-plot.png` (accessed Nov. 15. 2006)

The following figures are used subject to the GNU Free Documentation License 1.2:

- Figure 1.1 extracted from:
  `http://grotsnik.ogre3d.org/wiki/index.php/Image:Grass.png` (accessed Nov. 15. 2006)
- Figure 1.1 extracted from:
  `http://grotsnik.ogre3d.org/wiki/index.php/Image:Destina.jpg` (accessed Nov. 15. 2006)

The sound files for the sounddatabase are used from The Freesound Project[6], a collaborative database of Creative Commons licensed sounds and from Georg Holzmann's recordings: *KlangRausch*.

[1] `http://kile.sourceforge.net/` (accessed Dec. 12. 2006)
[2] `http://www.gnome.org/projects/dia/` (accessed Dec. 12. 2006)
[3] `http://www.fon.hum.uva.nl/praat/` (accessed Dec. 12. 2006)
[4] `http://audacity.sourceforge.net/` (accessed Dec. 12. 2006)
[5] `http://puredata.info/` (accessed Dec. 12. 2006)
[6] `http://freesound.iua.upf.edu/` (accessed Nov. 15. 2006)

Chapter 1

Introduction to sound textures

" The most obvious property of texture is perhaps its ubiquity. "

[Fang Liu]

Sound texture modeling is a widely used concept in computer music. Although, a concrete definition of sound texture is elusive, with this thesis I try to determine the scope of the different fields of "acoustic" texture and to group the various generation methods which can be put down from well-known techniques (e.g. wavelet transform, granular synthesis and source-filtering etc.) found in common computer music systems.

So far there has been a great interest in developing methods which generate sound textures of undetermined duration out of very short input textures only. Since most of the commercial applications (e.g. computer games) have a limited memory capacity, the focus of sound texture generation for these products is more on compressing the audio data than on sound quality. Scientific attempts of creating perceptually meaningful sound textures are not very common. Rather artistic approaches focus on the quality of sound textures but although real world sound samples hereby serve as orientation, realistic simulation is not necessarily the goal of these compositions.
In the practical part of this thesis two existing algorithms, named *audio texture* [Lu et al., 2004] and *natural grains* [Hoskinson and Pai, 2001], which are based on a similar granular analysis /resynthesis approach will be further investigated. Both algorithms can be used for creating a parametric sound texture generator that allows creating sounds out of a simple parameter control structure. Starting from a short input texture, different, new sound textures of variable length are produced. These new textures consist of consecutively connected patterns that are similar to the input texture.
In the Matlab programming environment the input parameters which make up the specific properties of the algorithm, such as frame size, number of feature coefficients, sequencing rules etc, are tested with respect to the resulting perceptual sound quality.
The implementation goal of this thesis is a real-time Pure Data (PD) higher-level interface for demonstration purposes allowing the user to manually adjust parameters until the produced sound texture sounds plausible and realistic with respect to a particular use of the texture. Ideally, the sound textures are not recognized as a resynthesized version, new sounds are considered as being natural, segments are concatenated in a natural flow and no disturbing repetitive patterns or artefacts can be heard.

The first part of the thesis (Chapters 1) embeds this work into the context of the various fields of texture generation and research. Chapter 2 looks at the current state of sound texture creation and related fields. In this chapter several approaches are presented and the problems of published sound examples are discussed.

The second part of the thesis (Chapters 3 to 4) describes the technical background and introduces details of the selected algorithms *audio texture* and *natural grains*. Furthermore, solutions for improving these algorithms are presented.

In Chapter 5 the real time sound texture generator implemented in PD is introduced, which incorporates the algorithmic improvements. In Chapter 6 general reflections upon perfect sound textures are presented resulting from investigations in this thesis. Finally, Chapter 7 discusses advantages and constraints of the algorithms and outlines some ideas for future work and research.

1.1 Texture

The curious thing about texture is that it cannot be associated with one single discipline. Texture (from Latin textura: *fabric* [Stowasser et al., 1994]) is a widely used term which characterizes consistence, structure and composition of complex items. Apart from the Latin denotation, *texture* can be found in several aspects of life.

Basically, it is important to mention that texture is a very language specific-term, that is especially common in English-speaking countries. Although it is a commonly used term in the English literature on music, it is particularly unclear what texture is supposed to mean in the context of music.[1] I shell now try to give a general idea of different domains and research areas where one comes across *texture*:

- **Texture script**: Textura or Gothic bookhand is a script that originated in North France in the Gothic. Nowadays this script form is associated with *Gothic* [Schumacher and Balthasar, 2006].
- **Crystalline texture**: The distribution of the grain orientations of polycrystalline forms refers to crystallographic texture [Ma et al., 2006].
- **Textural perception**: Texture is an important item of information that humans use for analyzing a scene [Amadasun and King, 1989]. Visual perception classifies texture in six categories, namely coarseness, contrast, directionality, line-likeness, regularity and roughness [Tamura et al., 1978].
- **Food texture**:

  > Perception of food is the result of food characteristics interacting with the processes in the mouth, as interpreted by the brain.[Engelen, 2004] (p 2)

  Food is usually described by the taste and the flavor. However, subconsciously the texture of food is of great importance for the appreciation and the recognition of food. In Figure 1.1 numerous factors are presented that are both product and subject related which influence food texture perception.

  The term texture is also used in anglophone language areas in cuisine and wineculture. In [Asimov, 2006] texture is explained by means of Champagne's effervescence which offers a different textural experience from that of most wines. Richard Geoffrey who is the cellar master from Dom Perignon Champagne explains "...you can feel the difference between a lively vibrant champagne and one that fatigues the mouth. Its texture!"
- **Image texture**: Image texture refers to surface. Everything that has a surface has a texture. The generation of image texture resembling a surface is an important field of research in computer graphics. Examples of image texture are shown in Figure 1.2 and an example of three dimensional image texture mapping is depicted in Figure 1.3.
- **Textures in composition**: [Dunsby, 1989] states that texture probably arose as a feature of the critical vocabulary spawned by post-tonal music starting in the beginning of the 20th century. It is an interesting matter of fact that the definition of sound texture is very open and differs from one approach to the next (compare 1.2.1), whereas different contemporary music composers such

[1] Dunsby who wrote an essay about music textures says: "It might [...] be asked what it is that English-speaking musicians have concerned themselves with while others have not.[Dunsby, 1989]"

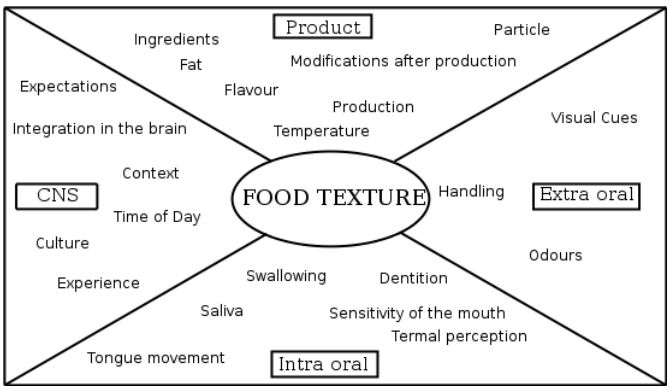

Figure 1.1: Diagrams of factors that can influence food textures. The abbreviation CNS stands for Central Nervous System. [Image adapted from [Engelen, 2004] .]

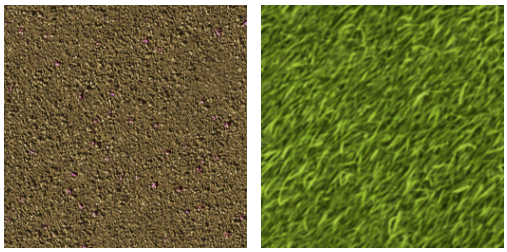

Figure 1.2: Examples of visual textures [Image extracted from [OGRE-team, 2006] under the terms of the public domain copyright.]

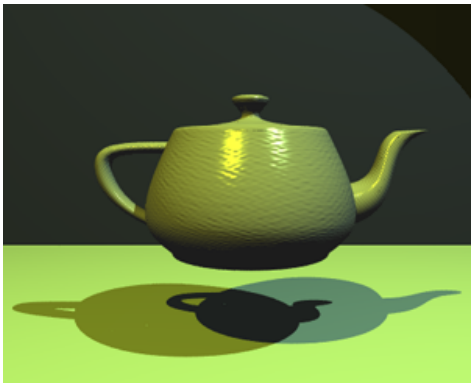

Figure 1.3: The Utah teapot (standard reference model in 3-D applications) on which a texture mapping algorithm is applied. [Image extracted from [Wikipedia/de, 2006] under the terms of the public domain copyright.]

as for example Lachenmann[2] have a very precise idea of what a (music) texture is [Lachenmann, 1996]. A detailed exploration into these textures goes beyond the scope of this thesis, since many elaborations (e.g. Iannis Xenakis: Concrete PH (1958) and György Ligeti: Volumina (1962)) would need to be examined in order to understand what is meant by textures in composition. Therefore, the reader is referred to [Bernadini, 2004; Dunsby, 1989] for further details.

- **Acoustic texture**: sound, music, sonic and audio texture. These terms are the main keywords of this thesis and will be explained further in the following sections.

As can be seen above there are several disciplines focusing on *texture*. What most of these fields have in common is that they all investigated some sort of material that can be explored with our human senses. From an ecological point of view[3] they all use real-world stimuli (visual system, haptic system, odour-taste system and auditory system) to explore the textural environment. In [Neuhoff, 2004] Valkenburg and Kubovy show the gap between types of modality quoting Gibson's illustration of the problem with his description of fire.

> A terrestrial event with flames and fuel. It is a source of four kinds of stimulation, since it gives off sound, odor, heat and light.... One can hear it, smell it, feel it, and see it, or get any combination of these detections, and thereby perceive a fire...For this event the four kinds of stimulus information and the four perceptual systems are equivalent. [Neuhoff, 2004]

1.2 Textures in the acoustic domain

1.2.1 What is a sound texture ?

Sound textures are an important class of sounds in interactive applications, video games, immersive virtual reality environments and web-based applications, movie sound effects, or in art installations. In video games it is important that sound textures can be used throughout the game without requiring too much diskspace. Furthermore, in an installation-based scenario the creation of a soundscape from a very short texture may be required.

As in image processing [Liu, 1997] there is no universally valid definition of a sound texture. Since the term sound texture is relatively new, several authors come up with *their* specific sound texture definition. This is sometimes very vague and spans from baby crying and horse neighing up to background sounds with simple musical structure [Athineos and Ellis, 2003; Behm and Parker, 2004; Cardle et al., 2003; Bernadini, 2004; Dubnov and N.Tishby, 1997; Dubnov et al., 2002; Scipio, 1999; Filatriau and

[2]Lachenmann describes five types of sound (German: *Klang*): *Kadenzklang*, *Farbklang*, *Fluktuationsklang*, *Texturklang* and *Strukturklang*. About the overall characteristic of texture he states:

> Betont sei nocheinmal die Tatsache, dass die Gesamt-Eigenschaft einer Textur nirgends mehr notwendig identisch ist mit den momentan darin zu hörenden Detail-Eigenschaften, allerdings in dem besonderen Sinn, dass der Komplexitätsgrad des resultierenden Gesamtcharakters, als oft eher statistisch zu bewertendes Resultat von Häufungen, meist geringer ist als derjenige der im Textur-Inneren eher beiläufig sich zusammenschließenden Gestalten - so wie eben die Masse meist primitiver ist als ihre einzelnen Komponenten. (*It should be pointed out once more, that the overall characteristic of texture is not at all necessarily identical with the detailed characteristics that one can hear in them at any given point in time, in the special sense that the level of complexity of the resulting overall idea (thought of as a statistical result of densities) is often lower than that of the implicit Gestalt forming itself from within the inside of the texture rather coincidentally - just as the multitude is often more primitive than the individual components are.*)[Lachenmann, 1996] (p 17)

[3]J. Gibson (1972) [Gibson, 1973] originally introduced the term *ecological perception*. He explained that what an organism needs from a stimulus, for the purposes of its everyday life, is obtained directly from invariant structures in the environment. Ecological psychology was developed primarily for visual perception but it was further explored into the auditory domain by N. Vanderveer (1979). Hence, also the field of ecological psychoacoustics emerged, which pairs two fields of the study of auditory perception (ecological perception and psychoacoustics).

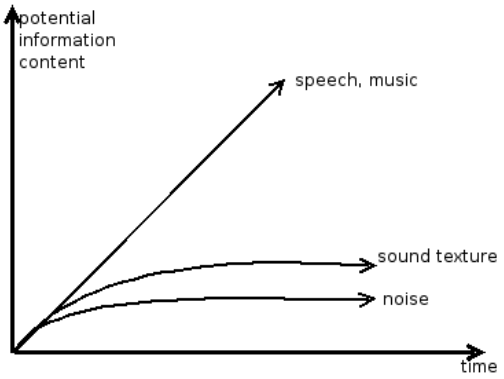

Figure 1.4: Sound textures and noise show long-term characteristics. Image adapted from [Saint-Arnaud and Popat, 1998]

Arfib, 2005; Misra et al., 2006; Hoskinson and Pai, 2001; Miner and Caudell, 2005; Norris and Denham, 2005; Saint-Arnaud and Popat, 1998; Wishart, 1996; Zhu and Wyse, 2004]. My thesis adheres to the definition from [Saint-Arnaud and Popat, 1998] who define sound texture using two major constraints: constant long-term characteristics and attention span.

> A sound texture should exhibit similar characteristics over time. It can have local structure and randomness but the characteristics of the fine structure must remain constant on the large scale. A sound texture is characterized by its sustain. [Saint-Arnaud and Popat, 1998] (p 294)

This definition implies that pitch should not change dramatically as, for instance an accelerating car, and rhythm should neither accelerate nor slow down.

> Attention span is the maximum time between events before they become distinct. High-level characteristics must be exposed within the attention span of a few seconds. [Saint-Arnaud and Popat, 1998] (p 298)

For this thesis especially the second constraint is very interesting regarding the point of how many events have to happen in order to be able to differentiate between one single event and a continuous stream of events which we denote as a texture. Illustrative examples include a single car sound versus the complex soundscape of traffic or a human voice versus a crowd of talking people (compare Section 3.1).

[Saint-Arnaud and Popat, 1998] also segregate sound textures by showing how the ”potential information content” of speech, music, sound textures and noise increase over time. According to Figure 1.4 speech and music provide new information at any time and their curve of ”potential information content” is shown as a continuously increasing function of time. Sound textures as opposed to speech and music have long term information characteristics. Their curve of information content becomes flat after a short time. Noise, in the auditory cognitive sense contains somewhat less information than sound textures.

Sound texture examples

When analyzing the examples of sound textures covered in most of the investigations on this subject, the following classes can be differentiated:
Natural sounds: fire, water (rain, waterfall, ocean) wind, surface impacts
Animal sounds: sea gulls, crickets, humming
Human utterances: babble, chatter

Machine sounds: buzz, whir, hammer, grumble, drone, traffic
Activity sounds: chip, sweep, rustle, typing, scroop, rasp, crumple, clap, rub, walking

[Dubnov et al., 2002] also include the sound of a crying baby. According to the above mentioned definition this sound should not be regarded as a sound texture, as the characteristics of the fine structure are not constant enough .

For the practical implementation of this thesis a sound database was created containing examples as listed above. In Chapter 6 it will be explained what results input sound textures produce when the signals are analyzed, split into segments and concatenated in a new order.

1.3 Typology of sound textures

While doing the literature research, I hardly found works presenting a profound typology about noise or sound texture classes, which I consider as a great scientific gap. Basically, [Hanna et al., 2004] state that automatic audio classification systems consider sounds as music, speech, noise or silence. According to [Filatriau and Arfib, 2005], from a perceptual point of view sound textures are good candidates to be described by the ecological approach initiated by Gibson [Gibson, 1973]. However, I found two typologies that fit in the context of sound textures.

1.3.1 Classes of noisy sounds

[Hanna et al., 2004] suggest a set of four classes of noisy sounds which are based on perceptual properties.

Colored noise

The first category covers the sounds that can be perfectly obtained by filtering white Gaussian noise. The sound examples cover sounds like a seashore, wind and breathing.

Pseudo-periodic noise

Several natural noisy sounds such as, for example humming insects or machine noise, are characterized by their pitch. These sounds are considered as a sum of a few sinusoids that imply noise with a perceived pitch.

Impulsive noise

The third category covers natural noisy sounds which are composed of periodic or aperiodic pulses such as applause, rain drops and walking. These pulses contained in impulsive noises are similar to transients (attacks) in instrumental sounds.

Sinusoids with noise

In the last class, real world sounds, such as street soundscapes with horns, wind in trees with singing birds, seashore etc., are considered. These sounds are assumed as being mixtures of several sound sources which may also be harmonic.

1.3.2 Composition of sound scenes

A very recent work of [Misra et al., 2006] is based on the notion that environmental sound scenes are composed of events and background texture. Their findings are intended for a specific software implementation (compare Section 2.1.7). Nevertheless, it is an intersting issue to examine how they define a ”sound scene”.

Deterministic events

According to the authors deterministic events are composed of sinusoidal components which are often perceived as pitched events such as a bird's chirp.

Transients events

are brief stochastic sounds such as footsteps and fire crackles.

Stochastic background

This is the residue remaining after the removal of the deterministic and transients parts. Sound examples of stochastic background include sounds such as wind, ocean waves or street noise.

1.4 Textures in music

1.4.1 Music textures

A music texture is defined as follows: From a short example clip of music an infinite version is generated by changing the order of the segments of the original signal [Jehan, 2005]. Due to the preservation of the rhythmic and the metrical structure the new music texture never seems to repeat. This approach is essentially based on a metrical representation (downbeat, meter, etc.) and on grouping the segments by similarity. Originally, *music texture* is inspired by *video texture* [Schödl et al., 2000].[4]

It should also be mentioned that in English speaking countries the term music texture is used in reference to the overall structure of a piece of music, the number of parts playing at once, the timbre of the instruments playing these parts as well as the harmony and the rhythmic structure used in a piece of music. The formal terms that are used, describe the relationships of melodies and harmony, for example monophony and polyphony.

1.4.2 Audio textures

Audio texture [Lu et al., 2004] is introduced as a means of synthesizing long audio streams from a short example audio clip. Examples of audio textures include screen saver sounds, lullabies, game music and background music (see 2.1.4 and Chapter 3 for more details). As opposed to music textures from [Jehan, 2005] the temporal rhythmic structure is ignored.

1.4.3 Sonic textures

The papers of [Filatriau and Arfib, 2005; Filatriau et al., 2006] introduce the term sonic texture. Although using a different term, I have the impression that sonic texture corresponds to sound texture since these papers have a similar literature basis as this thesis (compare [Strobl et al., 2006]).

[4] The sound examples from Jehan cannot be compared with music texture examples presented by [Dunsby, 1989].

Chapter 2

State of the art

> *"Meaningful sounds, however, vary in much more elaborated ways than merely in pitch, loudness, and duration. Instead of simple duration, they vary [...] in repetitiveness, [...] in regularity of rate, or rhythm [...]. "*
>
> [James J. Gibson]

2.1 Current methods

Sound texture generation is at the intersection of many fields of research such as signal analysis, sound synthesis, music information retrieval, sound design and computer graphics. In this chapter, I aim to present a survey of analysis/resynthesis and synthesis techniques dedicated to sound texture generation. Finally, in the second part of this chapter common problems of generated sound textures, which are published by some authors are evaluated.

2.1.1 Methods inspired by visual texture research

> There is a crucial difference in the way that humans use acoustic and light energy to obtain information about the world...For humans, sound serves to supplement vision by supplying information about the nature of events, defining the "energetics" of a situation. [Bregman, 1990] (p 36)

An increase in machine processing power in the last ten years led to the development of numerous methods like real-time 3D rendering, image modeling and special effects editing in visual computing for automatic generation of videos. For more in-depth introduction to visual computing the reader is referred to [Nielsen, 2005].

In order to create soundtracks for videos and computer games the demand of sonifying animated sequences came up. However, sound data is not analog to image data. Hence, sound texture approaches that have their background in image processing can never be a simple data mapping.

I think, it is interesting to tell that at the beginning of this work, I had the idea to transpose methods from computer graphics to audio. However, after doing literature research on sound textures, I decided to implement two algorithms that are signal-processing orientated.

Synthesizing sound textures through wavelet tree learning

The following presented approach is directly derived from prior work of the authors in texture movie synthesis [Bar-Joseph et al., 2001]: A statistical learning algorithm is presented for synthesizing sounds

that are statistically similar to the original [Dubnov et al., 2002; Bar-Joseph et al., 1999]. The main task in statistical learning is the estimation of an unknown stochastic source given one or more training examples, which are ”samples” from the source. A ”sample” can be a sequence of a movie frame, a texture image or a segment of a soundfile.
An input sound texture is decomposed into wavelet coefficients. Out of the wavelets the algorithm captures the joint statistics of the coefficients across time and scale. Thereupon, a multiple resolution analysis tree is created, which is used for the manipulation of sound grains that have a similarity to the original input texture. In the resynthesis step the inverse wavelet-transform is applied to obtain an output tree. By the use of a random generator the new grains are resynthesized. This approach is used for ”periodic” (ocean waves) and stochastic (crying baby) sound textures.

Creating audio textures by samples: tiling and stitching

[Behm and Parker, 2004] attempt to create sound textures of undetermined duration from short input textures . Starting from image processing, existing methods for creating visual textures, such as (*tiling*[1] and *stitching*[2]), are transferred to the sound domain. The tiling-based method uses a chaos mosaic[3] to generate a new sequence from a sound texture sample whereas the stitching-based method combines multiple chunks using a Least Recently Used (LRU)[4] algorithm.

2.1.2 Source-filter approaches

Sound texture generation methods based on source filter processing extract the spectral envelope from an input texture. The retrieval of the spectral envelope is based on an estimation of an all-pole synthesis filter. This approach is called Linear Predictive Coding (LPC), which is widely used in speech communication. In order to resynthesize the original signal, a white input signal is used as the source (excitation) signal that is filtered by the all-pole coefficients that are extracted from the original signal.

Two methods are presented, which aim at modeling sound textures by capturing properties of the excitation and the filter using linear prediction both in time and frequency domain. These methods are effective on texture types that primarily contain micro transients like crackling, crumpling or fire sounds, but do not work well on sounds that consist of different event types, such as for example environmental soundscapes with chirping birds [Misra et al., 2006].

Sound texture modelling with linear prediction in both time and frequency domains

Apart from music and speech signals, sound textures are considered as a third class of sounds [Athineos and Ellis, 2003]. In this work texture is modelled as rapidly modulated noise by using two linear predictors in cascade. The first linear prediction operation is applied in the time domain in order to capture the spectral envelope. The second linear prediction is carried out in the frequency domain using the residual of the previous LPC analysis to estimate the temporal envelope of the input texture. In the resynthesis step a filtered Gaussian white noise is used to feed the cascade of filters, which consist of coefficients that were obtained by the analysis of the original texture sample.

[1] An output image is tiled together from duplicates of the sample image, in a simple copy-and-paste manner [Nielsen, 2005].

[2] Stitching images is a method of merging multiple, overlapping images into one continuous image.

[3] For the creation of a chaos mosaic the following steps have to be executed: The output image is filled completely by tiling, which results into a repetitive image with visible seams. Consequently, randomly selected parts of random size of the sample are copied and pasted randomly onto the output image. Finally, the output image is filtered in order to smooth edges.

[4] The Least Recently Used page replacement algorithm is a cache algorithm, which discards the least recently used items first [Tanenbaum, 1992].

Sound texture modelling and time-frequency LPC

Similar to the approach presented by [Athineos and Ellis, 2003], [Zhu and Wyse, 2004] apply an extended time frequency LPC method to create new sound textures. The major goal is to synthesize arbitrarily long audio streams that are perceptually similar to the original sound. After the Frequency Domain (FDLPC) computation, the event density over the entire frame is calculated as a statistical feature of the sound texture and is used in the synthesis process to control the occurrence of events. In a further step the detected events are extracted, leaving a background sound devoid of any events. The individual segments are concatenated and a Time Domain (TD) LPC filter is applied to the background sound to model it. The obtained TDLPC coefficients are used to reconstruct the background sound in the resynthesis process. In a next step the time and the frequency domain LPC coefficients are clustered using k-means to reduce the amount of analysis data. In the resynthesis process the background sound and the event sequence are generated separately and mixed subsequently. A noise excited background filter is used to generate the background sound. Finally, in order to generate the foreground sound the event density number is used as parameter of a Poisson distribution to determine the onset position event in the resynthesized sound.

2.1.3 Wavelet/Filterbank-based methods

Analysis and synthesis of sound textures

[Saint-Arnaud and Popat, 1998] describe sound texture as a two-level phenomenon, having a low-level (atoms) and a high-level basis (distribution and arrangement of the atoms) (compare the sound texture definition from the authors in Section 1.2.1). In this implementation the input signal is analyzed using a Quadrature Mirror Filterbank (QMF). The input signal is split in six-octave wide frequency bands. Consequently, the energy level in each band of every frame make up a feature vector. A cluster-based probability model (k-means) that encodes the most likely transitions of feature vectors, is used to characterize the high-level of sound textures. Finally, the resynthesis is again performed by a binary tree structured QMF bank.

Stochastic based sounds for immersive environments

[Miner and Caudell, 2005] divide stochastic sounds in two basic classes: continuous sounds, such as motors, fan and wind and impulsive sounds, such as doors, gun firing and glass knocks. Their analysis/resynthesis approach uses wavelet decomposition. Analysis coefficients are transformed (manipulation of the high-and low-frequency parameters) according to the perceptual effects of the various models (rain, brook, footsteps etc.) Model parameters manipulation translates into a new set of wavelet coefficients. The synthesis employs an Inverse Discrete Wavelet Decomposition (IDWT).

2.1.4 Grain-based methods

Granular synthesis

In granular synthesis Roads defines a grain as a microacoustic event, which has a typical grain length between 1 and 100 milliseconds. The envelope of every grain is shaped by a specific amplitude envelope. A single grain serves as a building block for sound objects so that by combining thousands of grains over time a sonic atmosphere can be created [Roads, 2004]. There is a long tradition in the electro-acoustic music of splitting sound samples into portions and manipulating them to generate new sounds. Granular synthesis represents probably one of the oldest approaches in computer music to create texture like sounds. On purpose, I say ”texture like sounds” because with grain based methods not every type of sound texture can be (re)-synthesized and it is also not intended to create recognizable variations of the original signal.

Manipulation and resynthesis with natural grains

[Hoskinson and Pai, 2001] present a method for extracting parts of an audio signal in order to construct a similar signal of indeterminate length. The used natural sounds correspond with the sound examples from other investigations in sound texture modeling. A six-level wavelet decomposition is performed on windowed frames of the input signal so that feature vectors are obtained. With these vectors a Euclidean distance measure over four frames is performed in order to detect natural transition points. These points indicate parts where the least energetic changes in the signal appear. The sound in between these transition points is not broken up any further. Hence, these segments are called *natural grains*. For each segment a table of similarity between it and all the other segments is constructed. After the segmentation a first-order Markov chain is used where each segment is corresponding to a state of the chain. The transition probabilities from one state to the other are estimated based on the smoothness of the transition between it and all the other segments. Then the segments are arranged in a continuous stream with the next segment being chosen from the other segments, which best follow from it. There is a Java Applet[5] available that demonstrates this method.

Audio textures

[Lu et al., 2004] synthesize long audio streams using short audio clips as input signals. From an audio clip Mel frequency cepstral coefficients are computed in order to get a feature vector for every frame. A similarity measure is computed between any two frames. These measures are correlated with a kernel matrix in order to get a novelty score. Local peaks in the novelty correspond to the borders of the segments. Based on the transition probability between the extracted segments the sequences are concatenated in a new order. According to their definition audio textures includes sounds like lullabies, screen saver music and natural sounds (e.g. horse neighing, roar of the sea).

2.1.5 Synthesis methods

There are several filtering techniques colorizing white noise in order to create sound textures. However, in the context of sound texture generation based on sound synthesis only the work of Di Scipio based on Functional Iteration Synthesis (FIS) can be found in the literature. [6]

Synthesis of environmental sound textures by iterated nonlinear functions

Di Scipio [Scipio, 1999] uses nonlinear functions to synthesize environmental sound textures. The presented system is based on FIS, which is a derivative of the wave terrainy synthesis, where wave terrains are generated by iterations of nonlinear functions. Sounds like rain, cracking of rocks, burning materials etc. are considered as sonic phenomena of textural nature that are synthesized with FIS.

2.1.6 Physical modeling based methods

According to [Fontana and Bresin, 2003] sound synthesis based on physical modeling has its roots in paradigms for the generation of traditional instrument sounds. Therefore, a traditional synthesis model is specified by understanding the physical generation process and its effects on sound production. Recently, some models for sound textures were developed, which are physical-model based but focus on dealing with the cause that produces a signal and not with the effect [Rocchesso and Fontana, 2003].

[5] `http://www.cs.ubc.ca/~reynald/applet/Scramble.html` (accessed Nov. 15. 2006)

[6] I particularly investigated literature that describe "sound textures".

However, physical based models are always designed for very specific sounds only. No general sound texture model is possible with physical models and generated sounds are basically very ”pure and clean” meaning that there are no foreground sound events (eg. bird, car horn etc.).

Models for liquid sounds

The acoustic emission of single bubble sounds is identified as the fundamental mechanism for liquid sound production. [van den Doel, 2005] developed a sound synthesis model for a single bubble sound. Consequently, he applies a stochastic model on the bubble for the real-time interactive synthesis of complex sounds in order to generate liquid sounds such as produced by streams, pouring water and rain.

Crumpling sounds

Ecological sounds describe higher-level events such as crushing or walking [Fontana and Bresin, 2003]. Crumpling sound occurs whenever a source emission can be modeled as a superposition of crumpling events. For example the sound of aluminum cans that are crushed is a composition of single crumpling events. Generally, [Sethna and Houle, 1996] found out that crumpling paper emits sound in the from of a stationary process made of single impulses, whose individual energy can be described by Poissons's processes. Based on these findings [Fontana and Bresin, 2003] modeled ecological events like crushing walking and running, starting from an existing impact model that is superimposed by temporal stochastic characteristics. Moreover, they created real-time demonstration patches in Pure Data where the crumpling sound can be controlled and listened to.

Sound textures based on computational fluid dynamics

[Dobashi et al., 2003, 2004] developed methods for real-time rendering of aerodynamic sounds and turbulent phenomena, such as swinging swords and fire. Vortex sounds are simulated corresponding to the motions of fluids obtained by the simulation. First numerical fluid analysis is used to create the sound textures. Then the texture samples are used for rendering the vortex sound corresponding to the motion of the fluids. As these sounds do not only consist of vortices (e.g. combustion, reverb), a combination of synthetic vortex and recorded sounds is used.

2.1.7 Related work and applications

Tapestrea

Techniques and Paradigms for Expressive Synthesis, Transformation and Rendering of Environmental Audio (TAPESTREA)[7] is a novel software framework by [Misra et al., 2006] for analysis, transformation and synthesis of environmental sound scenes. A ”sound scene” is described as an environmental sound that is composed of foreground events and a background texture. Spectral modeling is used for extraction of deterministic events, transient events (Compare Section 1.3) are detected by time domain analysis and the background texture is generated by a wavelet tree learning algorithm similar to [Dubnov et al., 2002].

Instrumental gestures and sonic textures

[Filatriau and Arfib, 2005] construct a mapping between instrumental gestures and sonic textures. They developed the ”texture scratcher”, which is a digital music instrument employing a gesture-based exploration of visual space. It consists of a MAX/MSP adaption of the FIS algorithm presented by [Scipio,

[7] `http://taps.cs.princeton.edu/` (accessed Nov. 15. 2006)

1999]. Another application is the sonic fern [Filatriau et al., 2006], a graphical object generated by FIS that drives a sonic process at the same time.

2.2 Sound examples

From [Behm and Parker, 2004; Dubnov et al., 2002; Athineos and Ellis, 2003] listening examples were provided on the internet. In all cases only the resynthesized versions are available. Unfortunately an objective comparison is not possible because the sound files differ to a great extent concerning the sampling rate (8000-44100), the file length and the type of file.

The most important listening observations are:

- Without having any information tag the type of sound texture can be recognized.
- The synthesized textures contain audible repetitions: evident repetition of blocks (e.g. crowd [Behm and Parker, 2004][8]) and implausible accentuations that create an undesired rhythmical pattern (e.g. waves) [Dubnov et al., 2002][9]
- The important events are very well synthesized but the background sounds appear blurred (e.g. fire [Athineos and Ellis, 2003][10])
- In some sound examples gaps of silence can be heard, that make the textures sound unnatural and disturb the notion of homogeneity, which is typical for the original recording (e.g. traffic [Behm and Parker, 2004]).

Actually, these listening observations emphasize the core motivation of my thesis: the improvement of the acoustic quality of sound textures. Currently, no scientific approach can be found that offers a broad range of qualitative sound texture examples. Since several acoustic problems, as presented above, are evident, I try to put special strength on avoiding these points in my work (see Chapter 4).

[8] `http://pages.cpsc.ucalgary.ca/~parker/AUDIO/` (accessed Nov. 15. 2006)
[9] `http://www.cs.huji.ac.il/labs/cglab/papers/texsyn/sound/` (accessed Nov. 15. 2006)
[10] `http://www.ee.columbia.edu/~marios/ctflp/ctflp.html` (accessed Nov. 15. 2006)

Chapter 3

Insight in two grain-based algorithms

" [...] the still largely undiscovered world of the fabric of sound [...] "

[Jonathan Dunsby]

From the presented literature I selected two methods for being implemented namely *audio textures* [Lu et al., 2004] and *natural grains* [Hoskinson and Pai, 2001]. The audio texture approach is intended for signals with simple musical structure, such as game and screen saver music, whereas the natural grain approach focuses on environmental soundscapes, such as crickets, birds chirping and bubbling of brooks. Although, these methods are designed for different signals, they are both analysis/resynthesis approaches. They deal with the question of which parameters to extract from the sample, how to perform the best segmentation and how to do the best resynthesis in order to create a new sample longer in duration but with similar quality to the original signal (see Figure 3.1).

Since no sound is synthesized, theoretically these two approaches can be used with almost any input signal. It seems clear that these methods need a parametric adaption in order to work also on sound textures, since they are originally designed for divers signals. However, with these methods it should be possible to cover many types of sound texture because the sound output always consists of segments taken from the input signal. This is a great advantage as opposed to synthesis methods like FIS (2.1.5), physical modeling (2.1.6) or source-filtering approaches (2.1.2), which are based on modeling a complete new sound and are thus constrained to the production of one group of sound only.

This Chapter is divided into three major parts. The first part covers Section 3.1 to Section 3.2. In these Sections I try to introduce concepts and terms that are necessary for a comprehensive understanding of the algorithms. The second part is dedicated to a detailed introduction to the audio texture algorithm and accordingly the third part introduces the details of the natural grain algorithm.

3.1 Creating a natural stream

In both algorithms the term *stream* is of fundamental importance. [Lu et al., 2004] define audio textures "[...]as a means of synthesizing long audio *streams* according to a short audio input[...]" and [Hoskinson, 2002] explains that "[...]A technique is presented to facilitate the creation of constantly changing, randomized audio *streams* from samples of source material[...]".

As the handling of this term in these approaches is very open I would like to show the original meaning of Bregman, who introduced that term. According to [Bregman, 1990], a perceptual unit that represents a single happening is referred to as an auditory stream. A series of footsteps can form a single experienced event, despite the fact that each footstep is a separate sound and also a singer with a piano accompaniment can be heard as a coherent happening, despite being composed of distinct sounds.

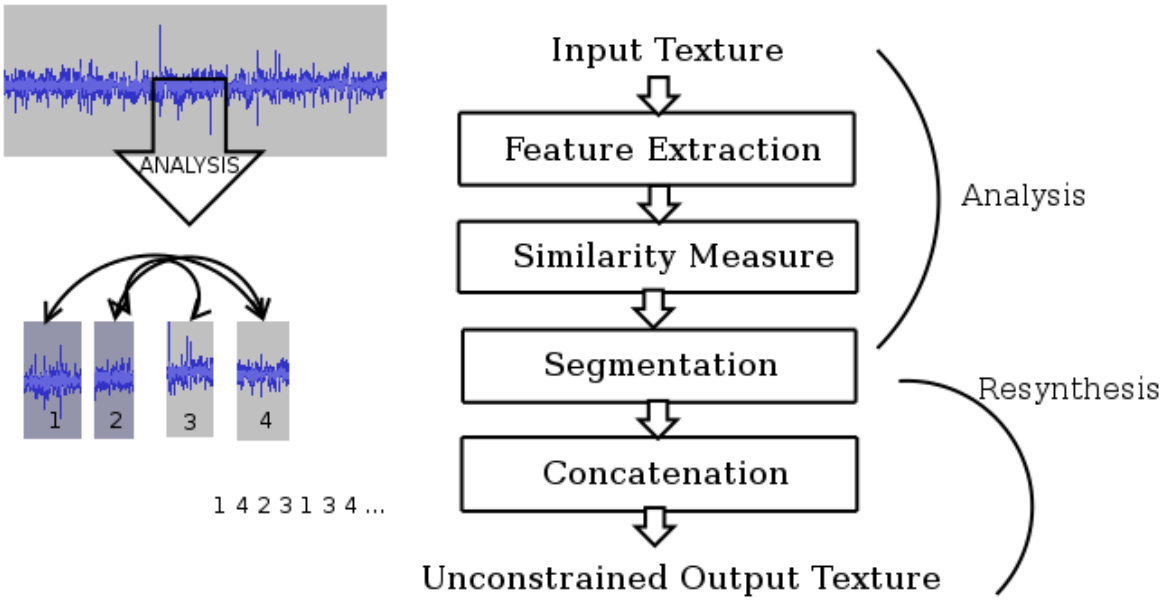

Figure 3.1: General algorithmic flowgraph for both *natural grains* and *audio texture*. Due to the analysis and the similarity measurements the original soundfile is split into *sub-clips*. Further computations calculate the similarity between any two sub-clips. Finally, a transition probability matrix is computed, which is used for generating a new sub-clip sequence order.

Bregman reserves the word stream for a perceptual representation and the phrase ”acoustic events” or the word ”sound” for the physical cause. Moreover, he states that a stream serves the purpose of clustering related sound qualities.

In this work the term ”natural stream of sound texture” is used very often. This should refer to the idea that even though the original signal, which consists of several short continuous streams, is split and concatenated in a new order, the resulting texture should still be considered as a continuous texture, which contains the same streams as the input sound file but arranged in a new order. It is also important that short streams within a soundfile, such as a car horn or a bird chirp, have to be preserved. In Figure 3.2 the spectrogram of a recording of sounds in meadow with lots of birds and insects can be seen. This special soundscape is made up of several streams (crickets, insects, bees and birds). Hence, it is important to detect all these streams as a whole and not to break them in between.

The adjective ”natural” is used to emphasize that the goal is not a simple concatenation of clips but a smooth sequence order of sounds that fit together and that support the perceptual notion of separate streams belonging together.

3.2 Feature vector

A feature vector is a n-dimensional vector of numerical features representing an audio file. Both methods are based on the calculation of short-time feature vectors. For each frame some form of audio analysis, Mel Frequency Coefficients (see Section 3.4.1) or Wavelet analysis (see Section 3.5.1), is performed and according to that analysis a feature vector for every frame is obtained.

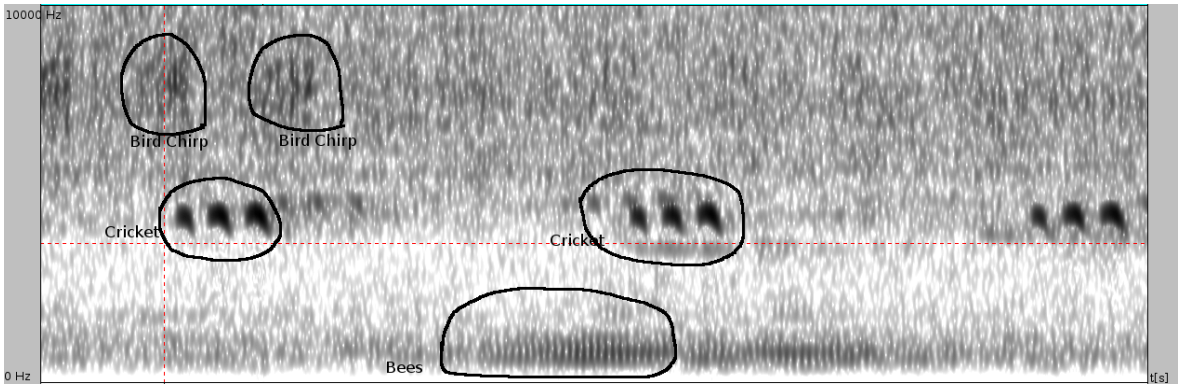

Figure 3.2: Spectrogram of several streams (bird and insect sounds) in the recording of sounds in a meadow. Without listening to the signal it can already be seen in the spectrogram that several streams build the entire sound and each visible group represents a single happening.

3.3 Distance and similarity measure

The basic idea of both algorithms is to find perceptually meaningful points in the signal where to cut. First the signal is analyzed. Owing to a special analysis a feature vector for every frame is obtained and consequently either a measure of distance or similarity is performed in order to find out how similar the frames are with each other.

The distance and the similarity measure can be used in order to quantify the similarity between two variables or two frames. Both measures investigate how close the distance is between two variables. Using a similarity measure, a strong similarity is expressed as a large value whereas the same fact is expressed by a small value from a distance measure. It can be concluded that a distance measure tries to quantify the dissimilarity between two variables [Brosius, 2002].

The detection of music similarity is a very common issue in music information retrieval and audio classification applications. Having a proper model of similarity enables automatic structuring and organization of digital music. The similarity data can be used for genre classification and play list recommendation [Flexer et al., 2005].

In Section 1.2.1 it was stated that a sound texture should exhibit similar characteristics over time. Since the audio texture and the natural grains approach are based on restructuring a sound file, similarity measures are performed in order to find out similar regions and transition points where segments can be combined in new order.

3.3.1 Distance measure

The Euclidean distance is a very common measure between two points. Moreover, there are other distance measures, which can be used as well, such as the Tschebyscheff, Manhattan and the Minkowski distance.

Euclidean distance

The Euclidean distance examines the root of square differences between coordinates of a pair of objects.

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{3.1}$$

Quadratic Euclidean distance

$$d(x, y) = \sum_{i=1}^{n} (x_i - y_i)^2 \tag{3.2}$$

A quadratic Euclidean distance is used for the natural grain approach.

3.3.2 Similarity measure

The cosine distance represents the most common similarity measure in music information retrieval. The similarity values range from *-1* to *1*. A similarity value of *1* reprents the highest similarity, meaning the distance of a variable with itself. On the other hand *-1* represents complete dissimilarity. Moreover, another measure that also ranges from *-1* to *1* is the Pearson correlation measure.

Cosine distance

$$d_{i,j} = \frac{\sum_{i=1}^{n} (x_i \cdot y_i)}{\sqrt{\sum_{i=1}^{n} (x_i)^2 \cdot \sum_{i=1}^{n} (y_i)^2}} \tag{3.3}$$

The cosine distance is applied for the audio texture algorithm to find out how similar frames are. .

3.4 Audio textures

The audio texture approach is derived from findings in rhythm analysis and music information retrieval [Foote, 1999]. In order to get a compact sound data representation for the similarity measurements Mel Frequency Coefficients (MFCCs) are computed for every frame.

3.4.1 Analysis using perceptual features: MFCCs

MFCCs are the most popular feature sets used in automatic speech recognition because they provide a concise representation of spectral characteristics [O'Shaughnessy, 2000]. [Logan, 2000] showed that MFCCs are also well suited for music analysis. Moreover, it is very common to use MFCCs in music information retrieval [Aucouturier and Pachet, 2004] when timbre similarity is measured. These coefficients provide a low-dimensional version of the logarithmic spectrogram, and thus are a good and compact representation of the spectral evelope. The process for computing the MFCCs is depicted in Figure 3.3.

The standard implementation comprises the following stages:

- First the signal is divided into frames by applying a windowing function at fixed intervals. Typically a Hanning window is used as windowing function, since it tapers towards zero and removes edge effects.
- For each frame a Short Time Fourier Transform (STFT) is computed:

$$S(e^j \omega) = \sum_{n=0}^{n} s_t(n) * w(n) e^{-j\omega n} \tag{3.4}$$

- The magnitude is converted to a logarithmic scale to get a dynamic compression because the perceived loudness of a signal is approximately logarithmic.

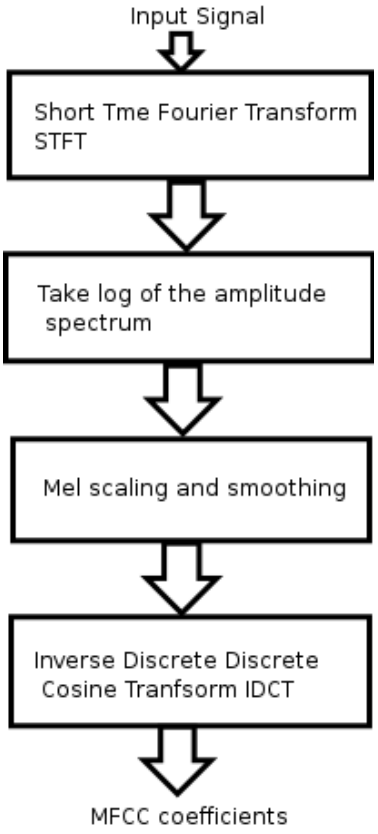

Figure 3.3: Computation of MFCC features. [Image extracted from [Logan, 2000]]

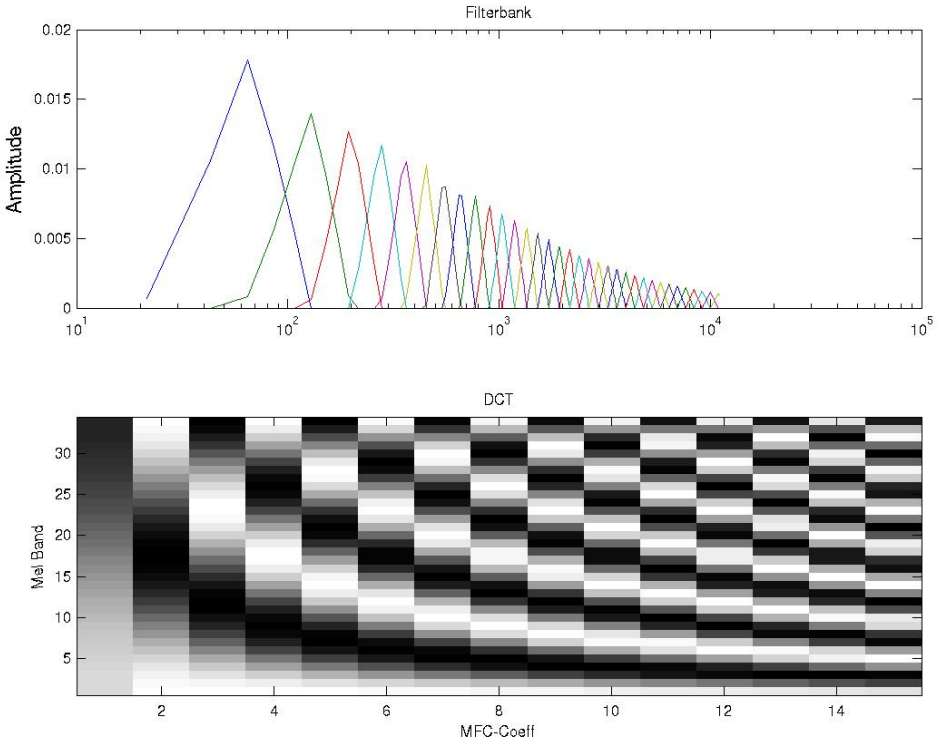

Figure 3.4: The upper plot shows the filterbank of triangular filters. The lower plot depicts the Discrete Cosine Transform matrix.

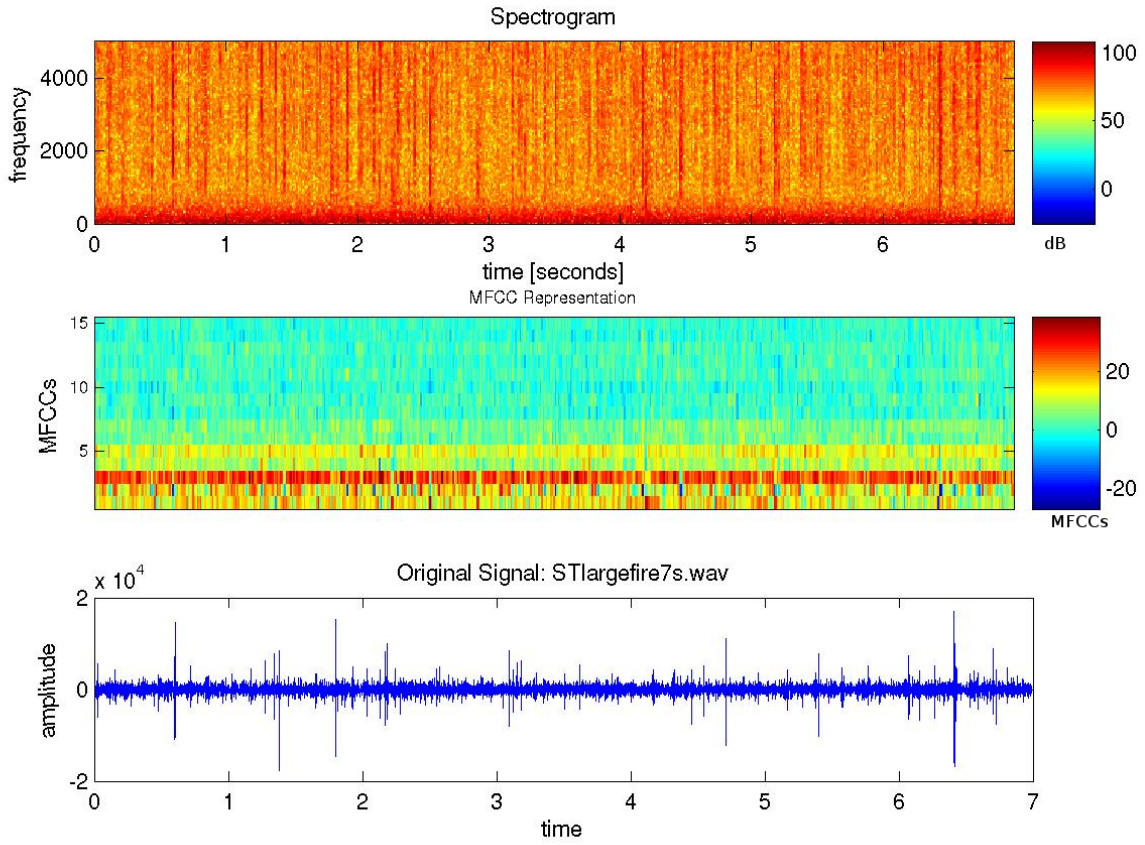

Figure 3.5: Spectrogram, RGB plot of the MFCCs and the original signal, 7 seconds of a recording of a fire sound.

- The next step is to smooth the spectrum and to emphasize perceptually meaningful frequencies. The perceptual weighting is based on the Mel Scale (See Appendix A.1 for details). The frequency bands of the filter bank can be seen in Figure 3.4.

- Finally, the Inverse Discrete Cosine Transform IDCT is computed in order to reduce the number of parameters. If *M* cepstral coefficients are desired they can be expressed as follows:

$$c_n = \sum_{k=0}^{N} X_k cos[n(k - \frac{1}{2})\frac{\pi}{N}] \qquad \text{for } n = 1, 2 \cdots, M \tag{3.5}$$

Xk denotes the log-energy output after the perceptual weighting and *N* represents the number of triangular filters.

The initial coefficient *c0* represents the average energy in each frame. *c1* reflects the energy balance between low and high frequencies. For $i > 1$, the coefficients *ci* represent increasingly fine spectral detail [O'Shaughnessy, 2000].

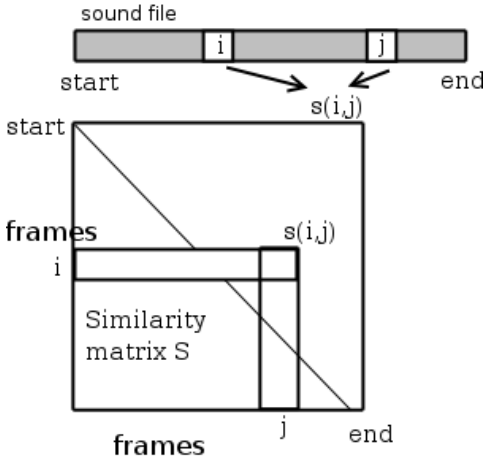

Figure 3.6: The panel depicts the embedding of the cosine distance measure between any two frames into the two dimensional similarity matrix. The similarity is an n×n matrix. *n* is the number of frames in the input texture. Image adapted from [Foote, 1999]

3.4.2 Frame similarity

After the computation of a MFCC vector for every frame the cosine distance between any two vectors is calculated.

$$s_{i,j} = \frac{x_i \bullet x_j}{||x_i||.||x_j||}^1 \tag{3.6}$$

$s_{i,j}$ represents the similarity between frame *i* and *j*. After the similarity between any two vectors is computed, the two-dimensional similarity matrix *S* is constructed (see Figure 3.6). According to [Cooper et al., 2006] the similarity matrix is a general method for visualizing musical structure via its acoustic self-similarity across time, rather than by absolute characteristics. This matrix contains the quantitative similarity between all pairwise combinations of frame vectors. The diagonal values of *S* are one, since every frame is maximally similar to itself and so the cosine distance is one. As can bee seen in the grayscale plots (see Figure 3.7, 3.8), where each pixel is given a greyscale value proportional to the similarity measure (the higher the similarity, the brighter the pixel), the diagonal is a white line. The brightness is proportional to the similarity. Similar regions appear bright whereas dissimilar appear in dark.

If a sound file, such as a cuckoo call, consists only of two extremely different sound objects, having two successive notes of different pitch, the simplified similarity matrix exhibits a 2×2 checkerboard pattern (see Figure 3.7) and can be defined as:

$$S = \begin{pmatrix} J & -J \\ -J & J \end{pmatrix} \tag{3.7}$$

J denotes a $n \times n$ all-ones matrix. The point where the notes change, corresponds to the center of the checkerboard. More generally, the boundary between two audio segments also generates a checkerboard pattern. The two segments exhibit high within-segment (self) similarity, producing adjacent square regions of high similarity along the main diagonal of *S*. The two segments produce rectangular regions of low between-segment (cross) similarity off the main diagonal. The boundary is the crux of this ”checkerboard” pattern.

Figure 3.8 shows the similarity matrix of a traffic soundscape (cars passing by) with an acoustically striking car horn. Since the car horn consists mainly of high frequencies the dissimilarity to the other frames can be clearly seen as a dark stripe in frames 75-80. Another interesting visualization is the

[1]The formula is the same as in Section 3.3, just the notation is simplified.

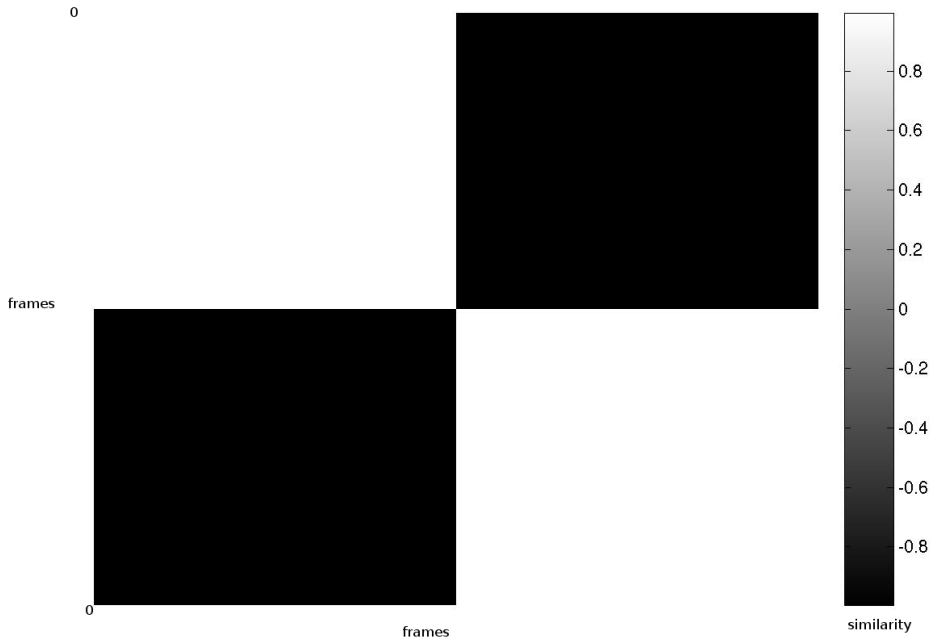

Figure 3.7: Ideal similarity visualization of two opposed sound objects.

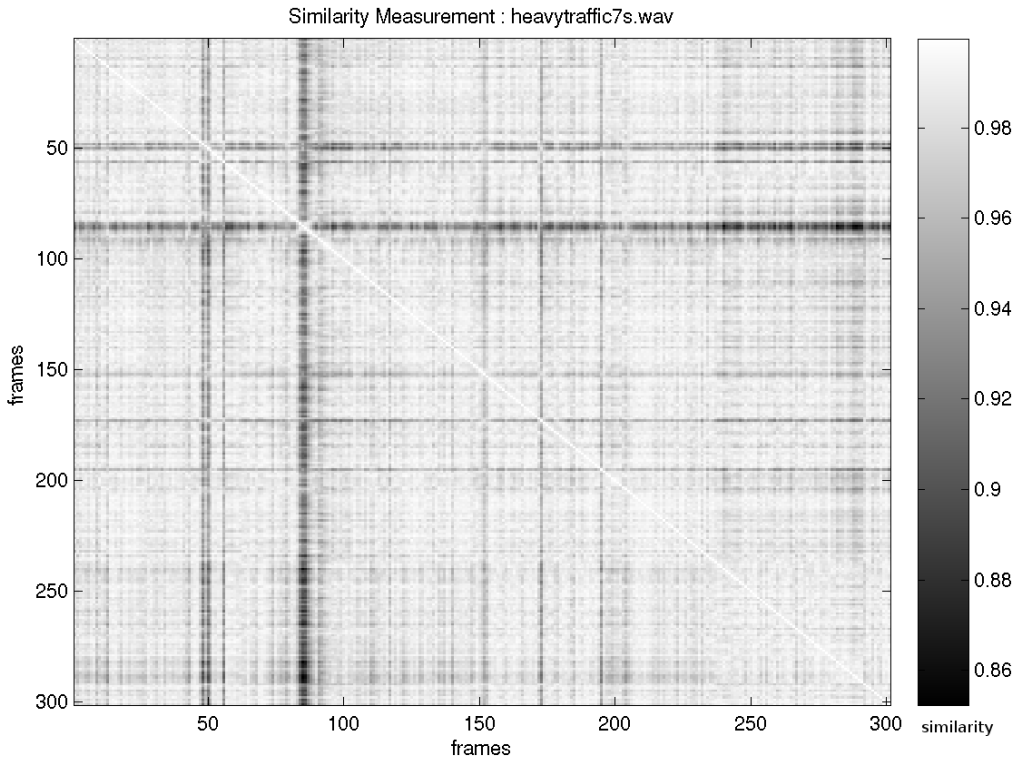

Figure 3.8: Greyscaleplot of a traffic signal with a horn.

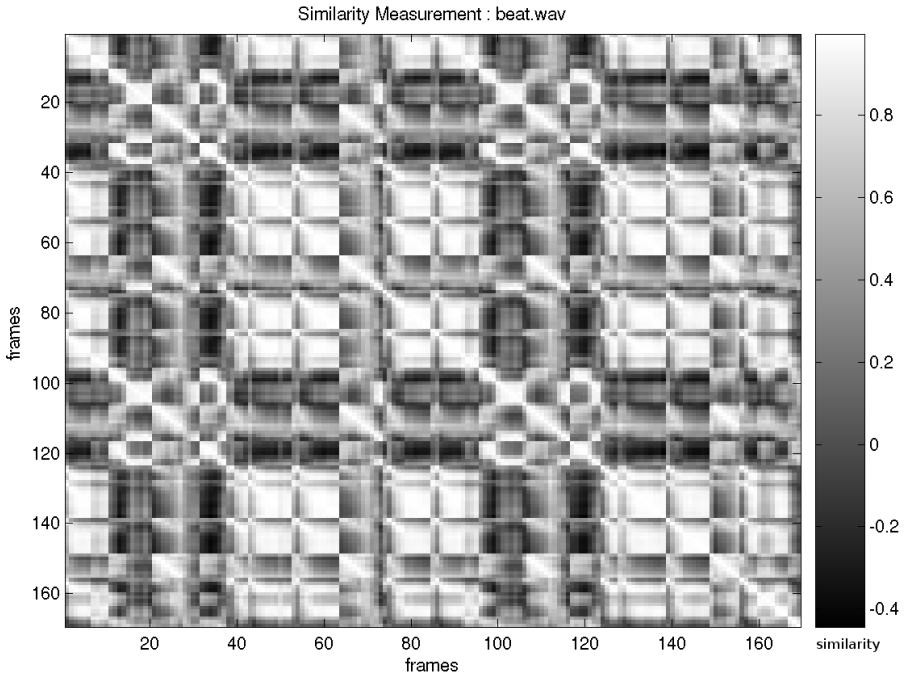

Figure 3.9: The rhythmic pattern can be seen very clearly. Time goes from left to the right as well as from the top to the bottom. Repeated themes are visible as diagonal lines parallel to the white main diagonal by the time difference between repetitions.

similarity matrix of a musical signal with a striking beat (see Figure 3.9). The rhythmic pattern is visible and the bright off diagonal stripes show repeated instances [Foote and Uchihashi, 2001].

3.4.3 Segmentation based on novelty-structure-analysis

> Novelty detection is the identification of new or unknown data or signal that a machine learning system[2] is not aware of during training. [Markou and Singh, 2003]

The objective is to select meaningful segments in the original signal. Segments are detected due peaks in the novelty score. Basically peaks represent frequency changes of the input texture. Novelty detection in this thesis refers to the detection of novel audio that corresponds to frequency changes. Therefore, S is correlated (see Formula 3.10 below) with a kernel[3] matrix in order to get the frame indexed novelty score. Peaks in the novelty score indicate audio segment boundaries. These local peaks are excellent candidate segment boundaries, because the audio is similar between maxima and significantly different across them.

To identify the patterns in S a "checkerboard" kernel is correlated along the main diagonal of the similarity matrix. A kernel matrix K basically looks like S but has a smaller dimension:

$$K = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \tag{3.8}$$

[2] Machine learning refers to a system capable of the autonomous acquisition and integration of knowledge. This capacity to learn from experience, analytical observation, and other means, results in a system that can continuously self-improve and thereby offer increased efficiency and effectiveness. [Hamilton, 2006]

[3] According to [Fisher et al., 2003] a kernel is a matrix of numbers that is used in image convolutions

K can be easily decomposed into a "coherence" and "anticoherence" kernel.

$$K = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{3.9}$$

The first term measures the self-similarity on either side of the center point. The second term measures the cross-similarity between the two regions. The difference between the two values estimates the *novelty* of the signal at the center point [Foote, 2000].

Peaks in the novelty score indicate locally novel audio, thus the correlation of *S* and *K* is referred to as a novelty score. The frame indexed novelty score *N* is computed as follows:

$$N_{(i)} = \sum_{m=-w/2}^{m=w/2} \sum_{n=-w/2}^{n=w/2} K_{m,n} S_{i+m,i+n} \tag{3.10}$$

w indicates the width of the kernel, which is centered at (0,0) [Foote, 2000]. Generally it can be said that the correlation process between *S* and the kernel is similar to an edge detection process in image processing. For computation with real signals the unitary matrix of the kernel is replaced by a two-dimensional Hamming window, which avoids edge effects because it tapers towards zero at the edges.

As mentioned above, local maxima correspond to perceptually significant changes in the soundfile and are thus being chosen as sub-clip boundaries. In Figure 3.10 it can be seen that local peaks of the novelty score correspond to the onsets of the rhythmical soundfile (compare Figure 3.9).

3.4.4 Sub-clip similarity

After the sub-clip extraction the similarity between any two sub-clips is calculated. Due to the novelty score the extracted sub-clips have a different frame count. Therefore, the similarity measurement is modified using a simplified time warping[4] method. If sub-clip *i* contains *M* and sub-clip *j* contains *N* frames and $M < N$, then the similarity between these two sub-clips can be expressed by

$$S'_{i,j} = \sum_{k=1}^{M} w_k s_{i+k,j+[kN/M]} \tag{3.11}$$

w denotes a symmetric rectangle window, which is used for frame weighting and $[kN/M]$ is the time warping factor.

In a next step the neighboring sub-clips are considered when the similarity between two clips is measured.

$$S'_{i,j} = \sum_{k=-m}^{k=m} w'_k S'_{i+k,j+k} \tag{3.12}$$

The transition probability from the *i*th sub-clips to the *j*th sub-clip is determined by $S''_{i+1,j}$. The more similar these two frames are, the higher the transition probability is. Consequently, the transition proability is defined as:

$$P_{ij} = Aexp(\frac{S'_{i+1,j}}{\sigma}) \tag{3.13}$$

A is the normalizing constant so that the the sum of each row in the the probability matrix *P* is unity. σ is a scaling parameter.

[4] Dynamic time warping is a method for measuring similarity between two sequences, which may vary in time or speed.

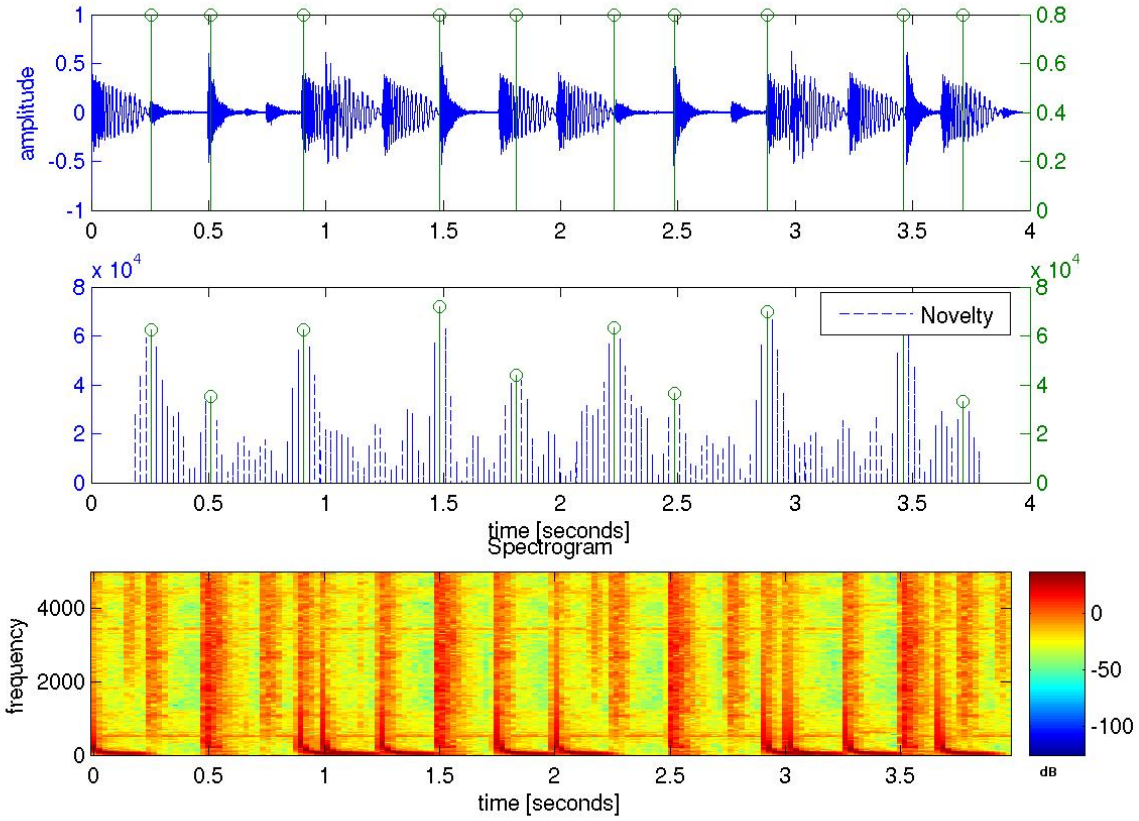

Figure 3.10: At the local peaks of the frame indexed novelty score, the signal is split into sub-lips. The plot depicts the novelty score of a rhythmic signal where the local peaks correspond to the onsets.

3.4.5 Sequence determination

The sub-clips are concatenated according to the maximum transition propability. Since $P_{i,i+1}$ always has the maximum probability the original sequence order would be kept. Thus, several constraints are defined in order to avoid the original sequence and repetition of sequences.

$$j \in \{j|P_{ij} > p_0\} \cap j \ni (l - r, l + r) \tag{3.14}$$

To introduce more stochastic variation in the generated sub-clip sequence, a random clip *j* is selected. The clip is only taken if its probability is bigger than a threshold p_0. Furthermore, the new selected clip is always tested whether it is in the adjacent range *r* of the previous sub-clip in the sequence. This constraint avoids the repetition of the original sequence order.

3.4.6 Concatenation of segments

The final output texture consists of segments taken from the input texture, which are concatenated in a new order. Therefore, in order to avoid clicks, the segments are concatenated using crossfades. A Hanning window is selected as a fading function since its amplitude ranges from 0 to 1, providing a smooth crossfade between adjacent sub-clips.

3.5 Natural grains

The algorithm is inspired by a work in speech segmentation from [Alani and Deriche, 1999] who segmented speech into phonemes using a six level wavelet decomposition. In contrast to the speech segmentation algorithm where energy maxima in the signal are detected in order to find phonemes[5], the natural grain algorithm focuses on the detection of natural transition points in the sample, *"[...]where events of some kind start or end[...]"*. Segments which cannot be broken up further are called "natural grains". Thus, the basic idea is, as opposed to the speech segmentation algorithm, to split the original signal into syllable-like[6] audio signals.

Once more the natural points are detected using a similarity measure. Now a wavelet decomposition is used to get a compact feature representation of the original signal.

3.5.1 Analysis using physical features

Wavelets are used because they produce a flexible signal decomposition and have the property of accentuating points in the signal where there are frequency changes. They can also be used to analyze the temporal and the spectral properties of non-stationary signals [Tzanetakis et al., 2001]. For every frame a six-level wavelet decomposition is computed and consequently for every level the energy of the difference coefficients is calculated. Therefore, every frame is parameterized as a six-level feature vector.

The Wavelet Transform

A wavelet (coming from the French *ondolette*, meaning "small wave") is a waveform with very specific properties, such as an average value of zero in the time domain and a limited duration. Wavelet analysis involves breaking up a signal into translated and scaled versions of the original (*mother*) wavelet to match the input signal. The Continuous Wavelet Transform (CWT) is defined as:

[5] A phoneme is the smallest contrastive unit in the sound system of a language [Grassegger, 2004].

[6] A word can be divided into syllables. Each syllable is a sound that can be said without interruption. Typically a syllable is made up of a syllable nucleus, most often a vowel, with optional initial and final margins, which are typically consonants [Grassegger, 2004].

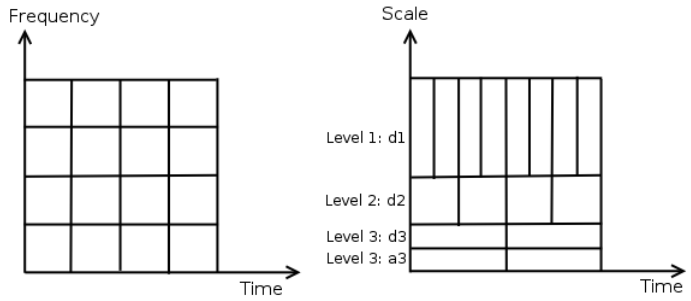

Figure 3.11: Comparison of the STFT and the DTFT; their different time and frequency resolution can be seen. d_n represents the detail coefficients and a_n the approximation coefficients.

$$W(a,b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} x(t)\psi(\frac{t-b}{a})dt \tag{3.15}$$

a is the scale factor, *b* is the translating factor and $\psi(t)$ is referred to as the mother wavelet. Common continuous wavelets are Haar, complex Mexican hat, Morlet and Hermitian [Strang and Nguyen, 1997].

The Wavelet Transform (WT) was developed as an alternative to the Short Time Fourier Transform (STFT) to overcome the problems related to its time and frequency resolution problems. The STFT provides a uniform time resolution for all the frequencies. In contrast to the STFT the Discrete WT (DWT) yields high time resolution and low frequency resolution for high frequencies and low time resolution but high frequency resolution for low frequencies. A comparison of the STFT and the WT can be seen in Figure 3.11.

However, the STFT and the DWT are linear transforms. The DWT is analogous to the CWT but uses discrete values for the scaling and the translating parameters.

The DWT can be seen as analogous to multirate filterbanks and as well related to a constant Q filterbank[7] with octave spacing between the centers of the filters. Each subband contains half the samples of the neighboring higher subband (see Figure 3.12). Hence, the signal is analyzed at different frequency bands with different resolutions by decomposing the signal into coarse approximation *a* and detail *d* coefficients. The coarse approximation is then decomposed further by successive highpass and lowpass filtering:

$$y_{high} = \sum_n x[n]g[2k-n] \tag{3.16}$$

$$y_{low} = \sum_n x[n]h[2k-n] \tag{3.17}$$

y_{high} and y_{low} represent the output of the highpass (g) and lowpass (h) filters, respectively after subsampling by a factor 2 [Tzanetakis et al., 2001].

3.5.2 Segmentation based on syllable-like audio segments

A six-level wavelet decomposition is performed on every frame of the original signal. Consequently, the energy is computed in each scale of every frame. After the computation of the feature vectors, a Euclidean distance function over four frames is used (see Figure 3.14).

[7] A constant Q transform is a bank of filters that have geometrically spaced center frequencies.

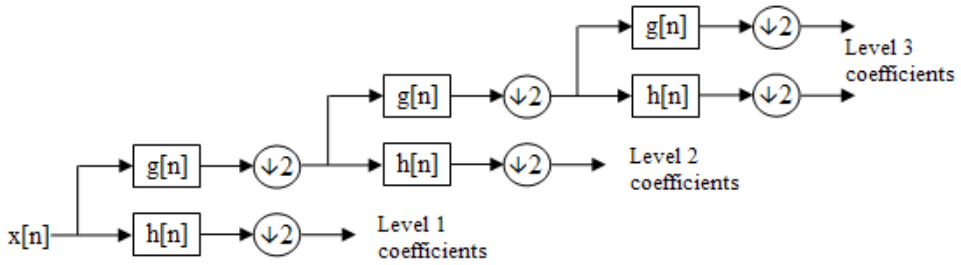

Figure 3.12: A 3 level filterbank [Image extracted from [Wikipedia, 2006a] under the terms of the public domain copyright.]

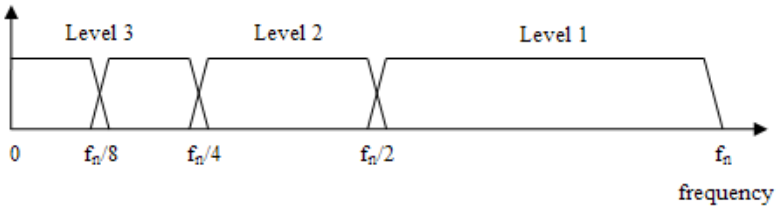

Figure 3.13: 3 Level frequency representation of the DWT [Image extracted from [Wikipedia, 2006a] under the terms of the public domain copyright.]

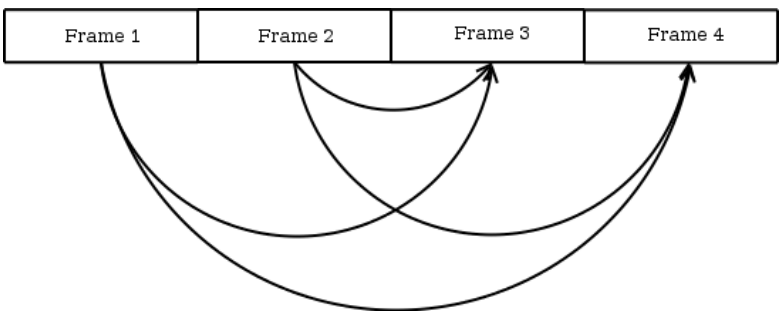

Figure 3.14: Distance measure for transition between frames 2 and 3. Image adapted from [Alani and Deriche, 1999]

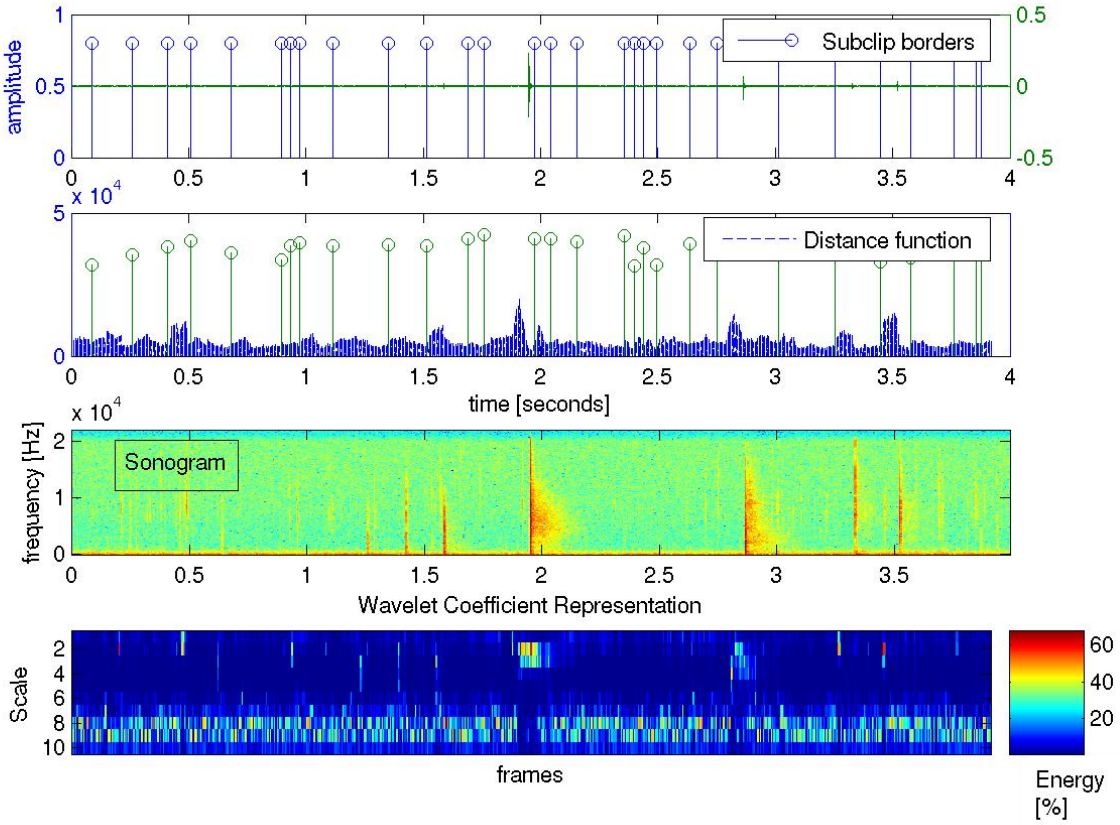

Figure 3.15: Euclidean distance function and local troughs were clips are segmented.

$$D(f2, f3) = \frac{1}{4}\sum_{i=1}^{2}\sum_{j=3}^{4}\sum_{k=1}^{m}(X_{i,k} - X_{j,k})^2 \tag{3.18}$$

$X_{i,k}$ and $X_{j,k}$ correspond to the energies of wavelet difference coefficients and m is the length of the feature vector. In order to show the correspondence between Figure 3.14 and the formula of the Euclidean distance over four frames, the sigma signs are dissolved, therefore D can be expressed as:

$$D(f2, f3) = \frac{1}{4}(\sum_{k=1}^{m}(\|X_{1,k}-X_{3,k}\|)^2+\sum_{k=1}^{m}(\|X_{1,k}-X_{4,k}\|)^2+\sum_{k=1}^{m}(\|X_{2,k}-X_{3,k}\|)^2+\sum_{k=1}^{m}(\|X_{2,k}-X_{4,k}\|)^2) \tag{3.19}$$

Thus, for every frame boundary a number is obtained representing how similar its neighbors are on each side.

The new sub-clips[8] boundaries correspond to the local minima in the frame indexed distance function. Too short sub-clips do not make sense for the auditory perception. Therefore, points are only detected when they lie beneath an adaptive threshold. Moreover, a minimum frame number defines the minimum number of frames in a subclip. If two minimum distance measure values appear in two adjacent frames, one of them is neglected, since the minimum frame size number does not permit short sub-clips.

[8]Natural grains are referred to as sub-clips in this thesis in order to use the same description of the extracted sound segments like the audio texture approach.

3.5.3 Grading the transitions and resynthesis

Now that the sub-clips are defined, the sub-clips are compared against each other to determine the most natural transitions. This is again done using a Euclidean distance measure. To calculate the Euclidean distance between two sub-clips, always the distance between the last two frames of a clip and the first two frames of another clip is computed. Out of these values a transition probability matrix is constructed. Since lower scores denote a smooth transition and high similarity between sub-clips, the inverse of the distance matrix $D_{i,j}$ is taken in order to orient the probability weights in their favor. $P_{i,j} = 1/D(i,j)$ indicates the likelihood that sub-clip *i* is followed by sub-clip *j*. In a further step the probability $P_{i,j}$ is converted to a probability $p_{i,j}$, where constant noise *C* is added to give values with smaller similarities more chance to be selected.

$$p_{i,j} = \frac{P_{i,j} + C}{\sum_{j=0}^{n} P_{i,j} + nC} \tag{3.20}$$

Finally, the probability matrix is normalized so that the sum in every row is one. The transition probability is used to construct a first-order Markov chain. Each state corresponds to a sub-clip. The next sub-clip to be played is chosen by a random sample of the probabilities and the smoother the transition between the current sub-clip and the next, the higher the probability that this sub-clip is chosen.

To conclude, the same crossfade resynthesis, as presented above in Section 3.4.6, is performed on the subclips.

Chapter 4

Algorithmic improvements

" Real-world sounds are complex, but they are also physically constrained. "

[W. A. Yost]

As the title of my thesis says, "parametric sound texture generator", it is obvious that the parameters for the generation of sound textures are very important. Therefore, the parameters composing the introduced algorithms are examined further in the following sections. Moreover, concrete solutions for parametric modifications are presented so that the quality of a sound texture is improved.

Special attention is paid to the improvement of the quality of the sound results, in order to create sound textures that can be used in practical applications. Repetitions should not be audible and sound textures should be targeted of sounding perceptually "meaningful", in the sense that the synthesized texture is perceptually comparable to the input texture. In the ideal case, no difference is noticeable so that the generated textures still sound naturally and comprise no artefacts.

The improvements of the algorithms especially focus on sound textures and sound scapes where a segmentation would also be possible with the segregation properties of our auditory system. Consequently, it is tried to determine perceptual meaningful segments and to create a sequence order that makes sense to our auditory perception.

I think, it is an interesting issue that both selected algorithms are derived from different signal processing areas. The audio texture approach uses findings from music information retrieval and rhythm analysis, whereas the natural grain algorithm directly adopts a method for speech phoneme detection. Once more, I have to emphasize that sound textures neither have the characteristics of music nor of speech signals. Therefore, it has to be examined how these algorithms have to be changed in such a way that they fulfill the requirements of sound textures.

The parametric improvements concern both the analysis of the input textures and the resynthesis of the sub-clips. First it is tried to find the optimal analysis parameters, such as for example framesize, number of MFCC coefficients or Wavelet decomposition level. Thereupon, it is investigated, which parameters influence the generation of the novelty score and the Euclidean distance function respectively and especially how these functions are analyzed so that single streams are fully covered in a segment (sub-clip). Finally, the generation of the output sequence is optimized so that repetitions can be avoided.

The testing software environment is Matlab[1]. For the computation of the MFCCs the MA toolbox from [Pampalk, 2004] is used.

The organisation of this chapter is presented almost in the same manner as Chapter 3. Sections 4.1-4.3 examine general aspects and improvements for both algorithms. Sections 4.4- 4.6 deal with the parametric improvements for the audio texture algorithm and Sections 4.7- 4.9 deal with the same for

[1] `http://www.mathworks.com/products/matlab/` (accessed Dec. 18. 2006)

the natural grain algorithm. Finally, Section 4.10 presents considerations of exchanging features and methods.

4.1 An empirical approach to encountering perfect segments

The algorithmic improvements that I am presenting in the following sections are all based on findings in signal processing research. However, the process of testing and finding the improvements was empirical. Only after listening to the (single) segments it was possible to start the modifications of the algorithms. Almost every parametric improvement in this work is the outcome of a concrete listening notion. Therefore, the working process was accompanied by listening permanently to individual segments, verifying for instance the spectrogram of the segment and changing a parameter if the result was not satisfying, according to the terms, defined in the introduction of this chapter. Usually textures that could rather be called soundscapes or textures including one distinct event (traffic environment with a car horn; compare Section 4.4.5) were the most helpful for findig deficits of the original algorithms. Taking the striking car horn as an example, the question was - how can we change the parameters so that the car horn is captured in a single segment? As a matter of course this is a very subjective task. Moreover, denoting a certain segment as ”disturbing” as done here (compare Section 4.5.1) might differ between individual listeners. However, I hold the opinion, that this empirical analysis/listening process is a specialty of my work, which provided new sound textures that can be considered as ”natural” as defined in the introduction of this thesis.

4.2 Sound database

In order to test the algorithms a sound database was created. As the main goal of this thesis is the improvement of the perceptual quality of the generated textures, I decided to use a high sampling rate of 44.1 kHz. The sample size is 16 Bit and the signals are mono. According to the sound texture groups presented in 1.2.1, sound files of 25 or 60 seconds are created in each case. The sound files consist of different recordings downloaded from the Freesound Project[2] and from the *KlangRausch*[3] recordings of Georg Holzmann. The sound files are available on the accompanied CD-ROM.

4.3 Frame size

In order to take advantage of the computational efficiency of the FFT algorithm that is included in the MFCC computation, the frame size N has to be a power of 2. For the Wavelet decomposition the frame size must be as well a multiple of 2^n, since n is the number of filterbank levels.
In this thesis several frame sizes are tested namely: 512, 1024, 2048 and 4096[4]. Although, the signals from the sound database are all considered as sound textures, they all have different properties. In order to test the long term characteristics of sound textures or soundscapes large frame sizes are used, which are ideally corresponding to the minimum time amount of sound that is necessary so that a particular sound stream (e.g. car horn) can be fully covered. Whereas for sound textures, which have a very fine structure like rain and fire short frame sizes are sufficient. A hop size of half the frame size is used for both algorithms.

[2] http://freesound.iua.upf.edu/
[3] A collection of noisy pitched everyday life sounds: car engine, coffee machine, fridge etc.
[4] In the real-time sound texture generator all these frame sizes can be tested for any input texture (see Chapter 5).

4.4 Improving the analysis: Audio textures

4.4.1 Using the parameters proposed by the authors

According to the Paper [Lu et al., 2004] the following parameters are used by the authors:

- Sampling frequency: 8-32 kHz
- Frame size: 256 samples
- MFCCs: 16 (no information about the use of 0th coefficient available) coefficients
- Duration of the original sound files: 2-15 seconds

In a first testing step these analysis parameters were used. As long as sound textures whose characteristic of the fine structure remained constant over time, the quality of the output texture was satisfying. However, as soon as sound textures were used that are composed of several streams, the parameters proposed by [Lu et al., 2004] did not produce convincing results (compare Chapter 5). Since the audio texture algorithm is not only determined by the parameters mentioned above, but also from others, such as for example the kernel size, I tried to investigate the impact of other components on the segmentation of the input textures.

4.4.2 MFCCs

When the MFCCs are computed the 0th coefficient represents the average energy in each frame. This means that including the 0th coefficient in the computation involves including signal energy values in the novelty score. The segmentation of the input texture results from local peaks in the novelty score. Thus, large energy values would produce peaks in the novelty score. Since the detection of frequency and timbre changes is more important than energy changes (amplitude), the use of 0th coefficient is neglected.

For music genre classification [Tzanetakis and Cook, 2002] propose the use of 5 MFCCs (at a samplingrate of 22,05 kHz)[5], according to the review of [Aucouturier and Pachet, 2004] in timbre similarity measurements the MFCC parameters vary between 8 and 20 and in automatic speech recognition it is common to use between 8 and 14 coefficients [O'Shaughnessy, 2000]. Sound textures have a very broad frequency range therefore the number of 16 coefficients, as proposed by [Lu et al., 2004] is kept.

4.4.3 Using a larger kernel size

According to [Foote, 2000] the width of the kernel directly affects the properties of the novelty measure. A small kernel detects novelty on a short scale only. Whereas large kernels average over short-time novelty and detect longer structures such as musical transitions and musical keys. Sound textures are not musical signals. However, after testing different frame sizes and kernel sizes, I found out that only a kernel having a size of at least 13×13 is able to detect single streams in the sound texture examples. This is easy to explain since a larger kernel implicates a correlation over more frames. Furthermore, the novelty score gets smoother when the kernel size is large, which makes it easier to detect local peaks.

In Figure 4.2 a part of a traffic signal with a striking car horn can be seen. Because of a 5×5 kernel the car horn is not detected as an entire event. It is cut into two pieces. The consequences of a small kernel can be heard in the resynthesis. "Two car horns" will appear, which disturb the homogeneous notion of a traffic environment. As a comparison the detected segment borders, which result from a 13×13 two-dimensional Hamming kernel can be seen in Figure 4.3. Now the car horn is detected as

[5] From the other authors the samplingrate is not known.

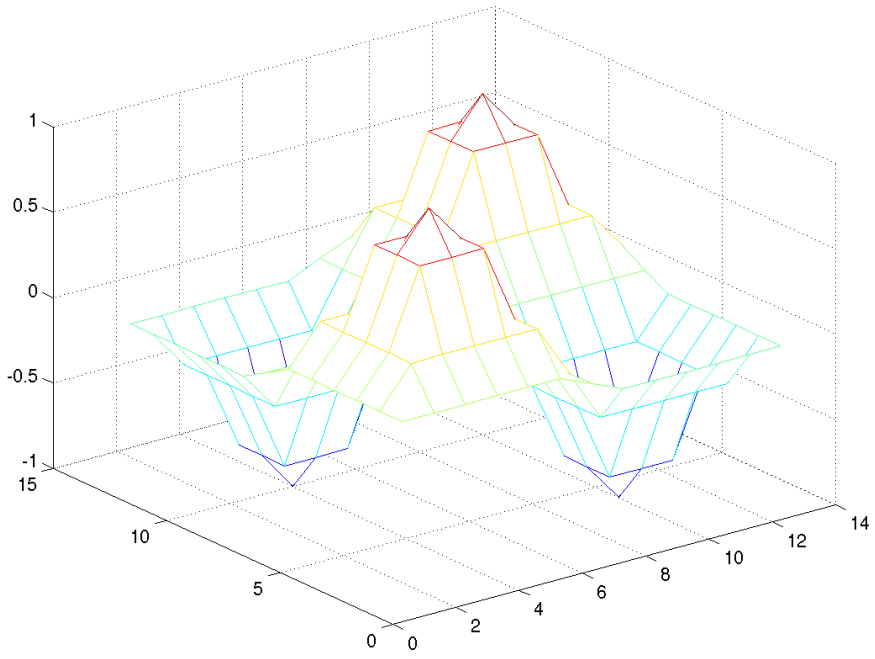

Figure 4.1: Two-dimensional Hamming kernel centered at (0,0).

a single event. Another example is shown in Figure 4.4[6] and Figure 4.5 where the effects of different kernel sizes can be seen on the segmentation of a bird chirp. Also in this case it is obvious that using a large kernel facilitates the detection of an entire stream. A 13×13 two-dimensional Hamming kernel looks like Figure 4.1.

4.4.4 Getting better segments

The sub-clip borders correspond to the local peaks in the novelty score. The local peak detection algorithm is dependent on two input parameters: the input signal (novelty score) and a ”delta” variable, a number between 1 and 9 in oder to get rid of small peaks around the local peak; the number refers to frames.
When listening to the resynthesized versions often very clear environmental background signals lose their natural smoothness due to different clip lengths and due to very short segments, which have their origins in small local peaks in the novelty score. Even though the peaks are correctly detected, if the frequency or energy change of the signal at some points is not strong enough the resulting sub-clip borders do not make sense to the auditory perception.
Testing the peak detection with a Gaussian white noise, which is interrupted in-between by a 440 Hz-sine, I discovered, that there are several detected peaks in the novelty score of the white noise signal, which are unwanted. Ideally three segments are expected: the white noise, the sinusoidal part and again the white noise. To get these logical segment borders a threshold in the peak detection algorithm is introduced so that local peaks are only used as segment borders if they exceed a certain threshold (a value between the minimum of all the local peaks and the absolute maximum of the novelty score). Using peaks above a certain threshold emphasizes that only points of strong spectral changes in the signal are used as new segment borders.

[6]The successive plots are snapshots from Praat. Gerhard Eckel developed an awk script that transforms the segment borders computed by Matlab to the required Praat format *.TextGrid (See Appendix A.2.3).

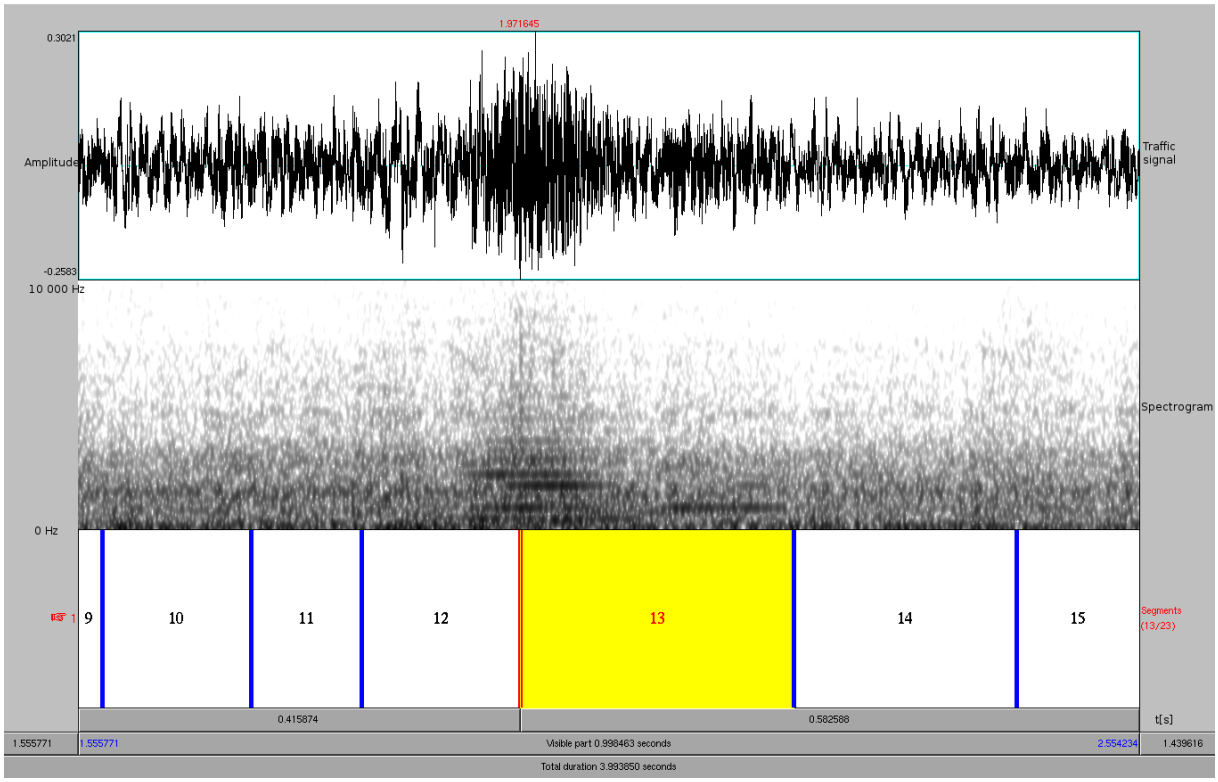

Figure 4.2: Traffic signal with a 5×5 kernel using a frame size of 1024. The car horn is split in the middle in such a way that it appears in sub-clip 12 and 13.

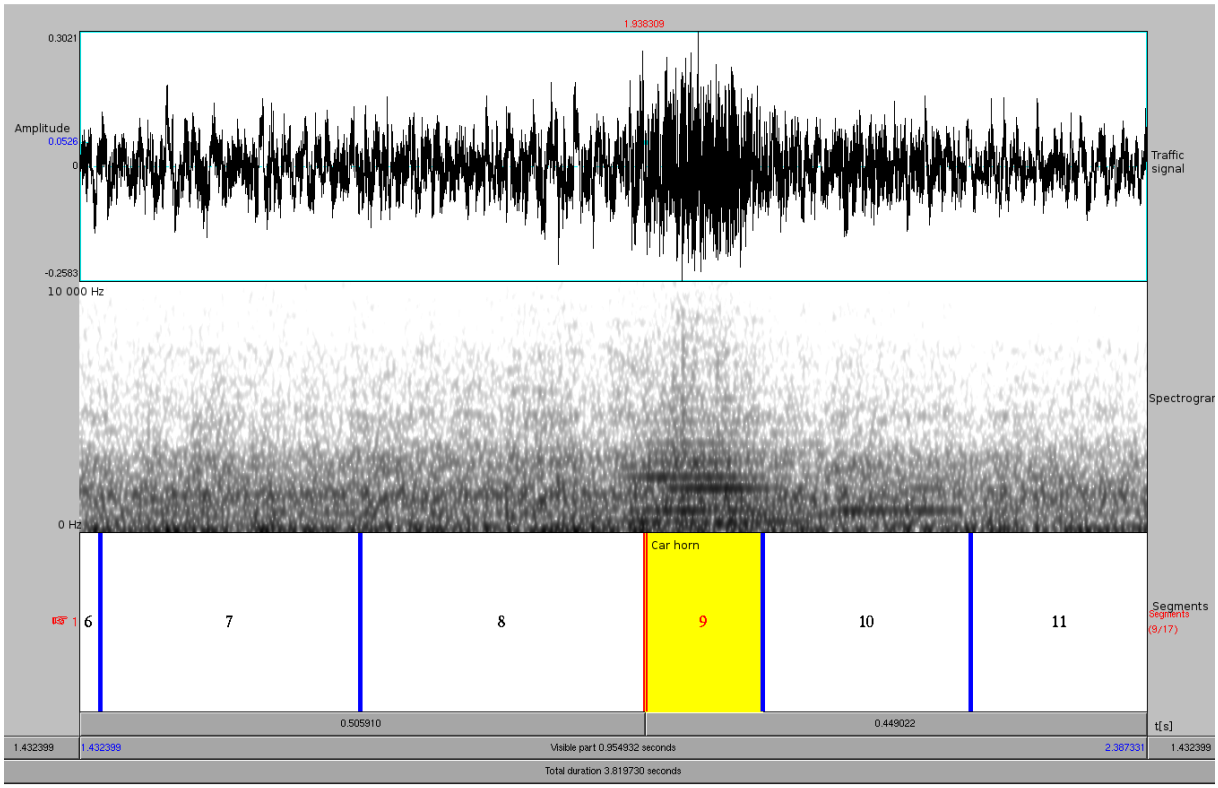

Figure 4.3: Traffic signal with a 13×13 kernel using a frame size of 1024. The car horn is detected as a single event and can be found only in the sub-clip number 9. Furthermore, it is visible that a larger kernel size produces larger and less sub-clips

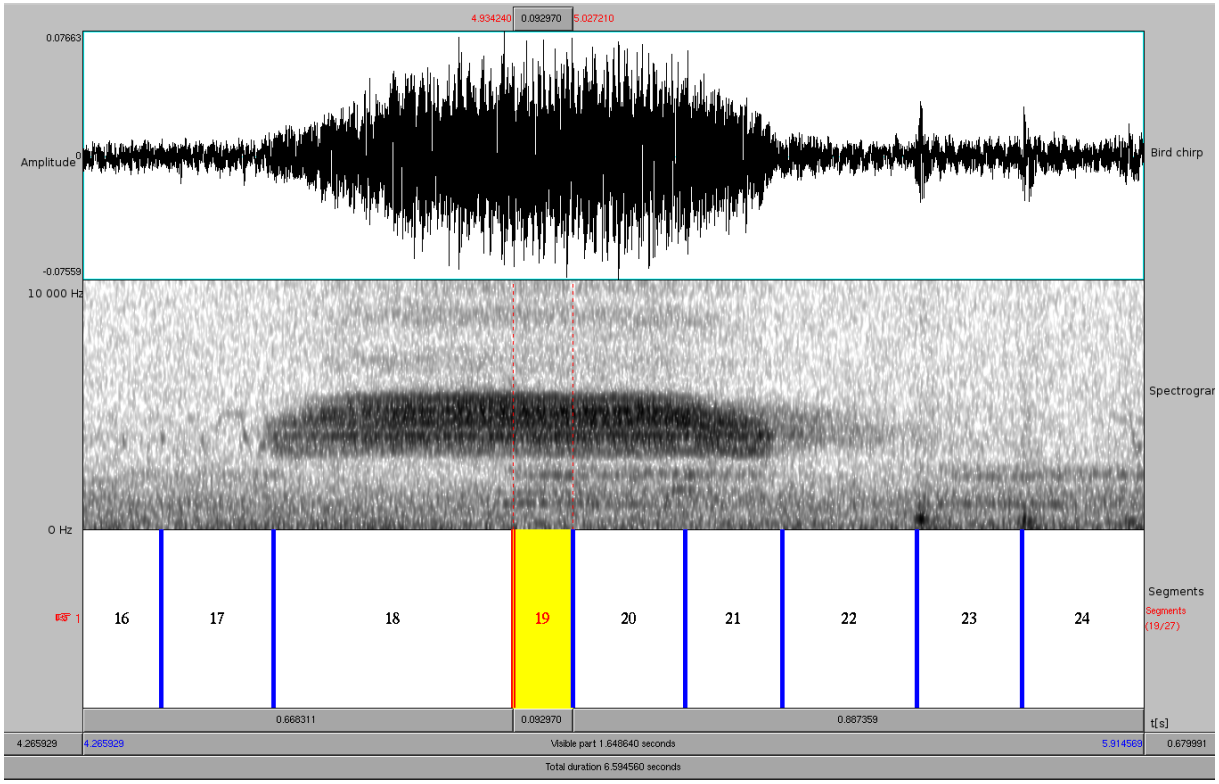

Figure 4.4: Bird chirp with a 5×5 kernel using a frame size of 1024. The chirp is cut into four sub-clips (see segment 18 - 21).

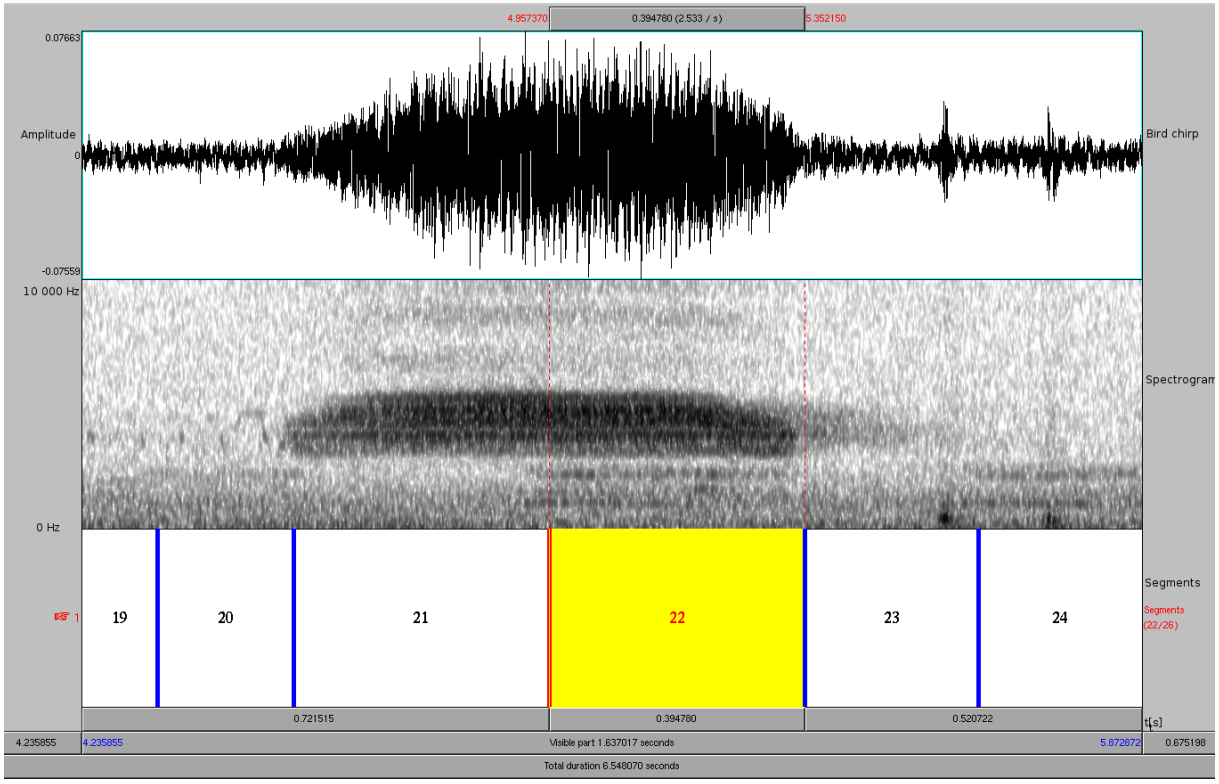

Figure 4.5: Bird chirp with a 13×13 kernel using a frame size of 1024. The chirp is cut in two clips only (see segment 21, 22)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%transfer the peaks of the novelty score->to peaks from original signal%
%---from frames---> to samples
[locMax, value] = localPeak(Novelty,delta);

[val, ind] = max(value); % find the maximum of the local peaks
[valm, indm] = min(value); % find the minimum of the local peaks
for i = 1:length(value)
    if value(i) <= (factor*(val-valm))+valm % check if the local peak is above a
        threshold
        locMax(i) = 0;
        value(i)  = 0;
    end
end

value  = nonzeros(value);
locMax = nonzeros(locMax);


Peak = zeros(1,length(locMax));

for i = 1:length(locMax) %index of a local Maximum
    Peak(i) = hop size*locMax(i) + 1 + fft_size/2;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
]
```

Listing 4.1: Matlab code for transferring peaks in the novelty score to the peaks in the original signal.

Before converting the peak values of the frame indexed novelty score back, to the input signal, the hop size has to be considered [Zölzer, 2002]. The Matlab code for the peak detection and for transferring the peaks (frame indices in the novelty score) to samples in the input texture, looks like Listing 4.1. *locMax* indicates the frame index of a local maximum and *factor* is a value between zero, and one, scaling the threshold. Due to windowing, the maximum is always in the middle of a frame, therefore always half the frame size is added to the computed sample number.

4.4.5 Enhancing the similarity measure between sub-clips

Since the extracted sub-clips never have the same length a time warping has to be performed for finding an optimal match between two given sub-clips. Once again, I would like to present the formula that incorporates the warping (compare Section 3.4.2):

$$S'_{i,j} = \sum_{k=1}^{M} w_k s_{i+k,j+[kN/M]} \tag{4.1}$$

w represents the rectangle window, which is used for frame weighting, $[kN/M]$ is the time warping factor and *M* or *N* respectively indicate the lengths of the sub-clips (*i* and *j*). If the sub-clips have the same length (the similarity of a sub-clip to itself) no warping has to be applied.

$$S'_{i,j} = \sum_{k=1}^{M} w_k s_{i+k,j+k} \tag{4.2}$$

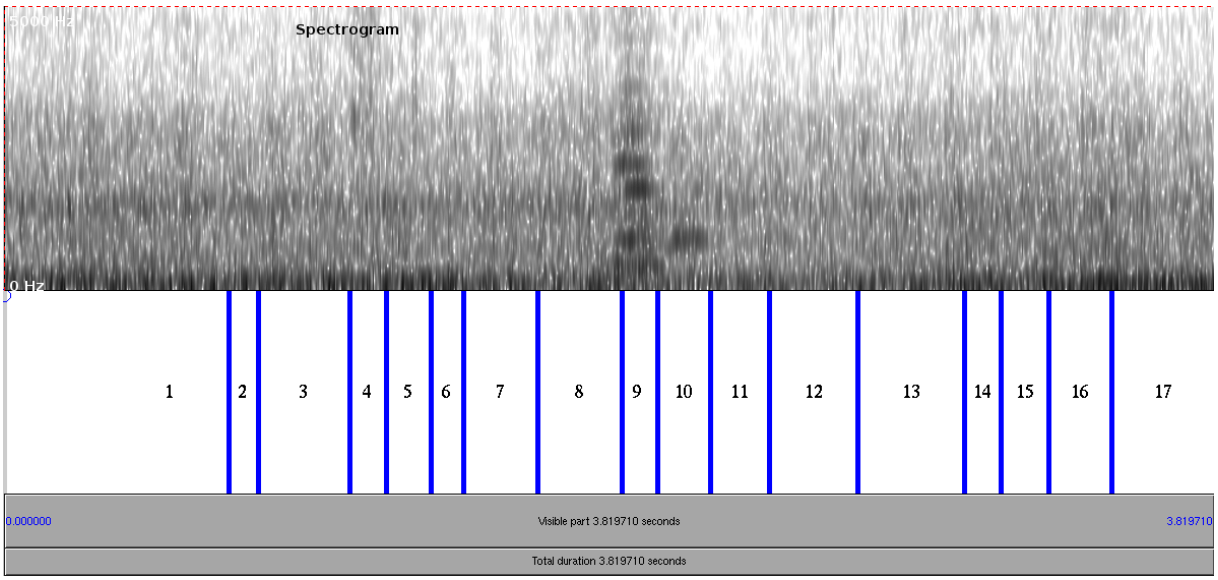

Figure 4.6: Spectrogram of a traffic signal with a car horn (segment 9). Sub-clip 9 and 4 contain frequencies that are different from the other segments. The displayed spectrum ranges from 0 to 5000Hz.

Definitely it can also be the case that sub-clip *i* contains *M* and sub-clip *j* contains *N* frames and $M > N$, then the similarity between these two sub-clips can be expressed by

$$S'_{i,j} = \sum_{k=1}^{N} w_k s_{i+[kM/N],j+k} \tag{4.3}$$

The problem of the above cited formulas is that owing to the summing up over *k* frames the similarity matrix does not have values, which range between -1 to 1. Thus, $S'_{i,j}$ does not define a cosine distance between two frames anymore but the sum over *k* (cosine) distances. Consequently, when a sub-clip is compared to itself the diagonal of the matrix now consists of the number of frames within a sub-clip and is not one anymore. This leads to the problem that the distance computation of a short sub-clip with another short sub-clip produces a small distance value, whereas a large sub-clip with another large sub-clip produces a large value. From the frame similarity the transition probability is computed. Owing to the frame summation in the sub-clip similarity matrix the transition probability matrix gets heavily distorted. According to formula 4.1 longer sub-clips automatically have a higher probability and short clips are denoted very dissimilar. In Figure 4.6 the spectrogram and the sub-clips of the traffic signal with the car horn can be seen. Sub-clip 9 contains the car horn and sub-clip 4 contains another event that is different from the other sub-clips. Although, these two sub-clips are different, this ”property” - the dissimilarity - cannot be seen in the similarity matrix anymore (see Figure 4.7).

Therefore, I propose the introduction of an averaging factor *1/M* or *1/N* respectively so that the formulas for computing the similarity between clips look like the following:

$$S'_{i,j} = \frac{1}{M} \sum_{k=1}^{M} w_k s_{i+k,j+[kN/M]} \qquad M < N \tag{4.4}$$

$$S'_{i,j} = \frac{1}{M} \sum_{k=1}^{M} w_k s_{i+k,j+k} \qquad M = N \tag{4.5}$$

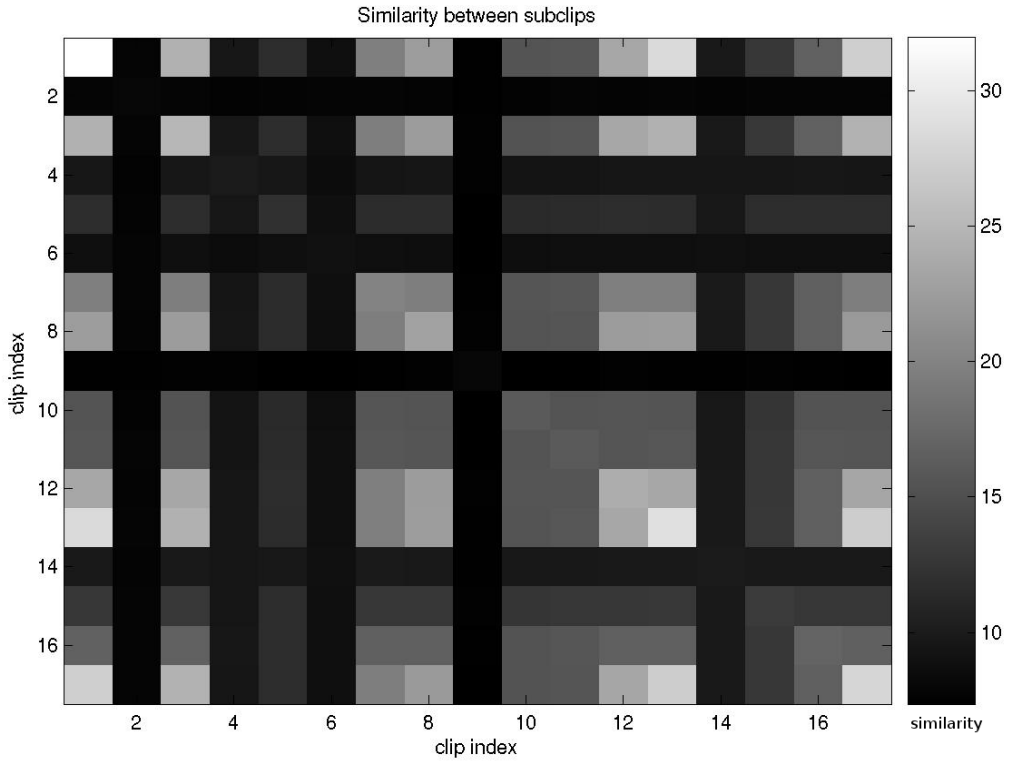

Figure 4.7: In this plot it cannot be figured out which sub-clips are dissimilar from the others.

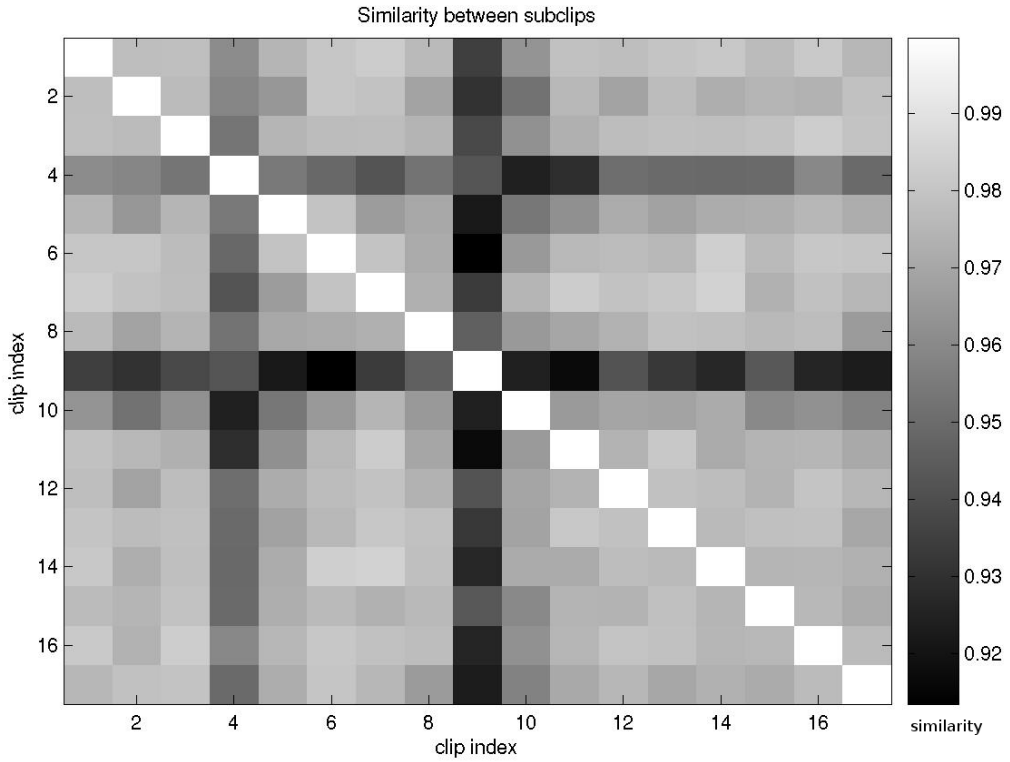

Figure 4.8: Due to the introduction of an averaging factor, dissimilar sub-clips (4 and 9) can be seen very well.

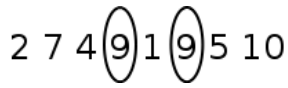

Figure 4.9: Clipnumber 9 succeeds shortly after one intermediate clip again.

$$S'_{i,j} = \frac{1}{N}\sum_{k=1}^{N} w_k s_{i+[kM/N],j+k} \qquad M > N \tag{4.6}$$

Now the self-similarity is again expressed as a diagonal vector of *S'* that is one. Moreover *S'* resembles *S* from Section 3.4.2 and the improvement is depicted in the grayscale plot of Figure 4.8. The values range between *-1* and *1*, the diagonal is one (white) and the two sub-clips that are different from the other clips appear, as expected, dark.

4.5 Improving the resynthesis: Audio textures

4.5.1 Modification of sub-clip sequencing rules

When testing the sound files from the database one major observation was, that when very continuous, fine grained sound textures such as fire crackling, liquid or noisy kitchen sounds were used, no repetition of the sub-clips in the resynthesis was noticeable. However, as soon as sound textures were used where the events that make up the texture can be sorted out and remembered from our auditory perception, unnatural repetition could be noticed. Due to these observations the constraints for successive sub-clips need to be stricter in order to avoid audible sub-clip repetitions when sound textures are constant on a large scale. Therefore, I also suggest to compare the new selected sub-clip with sub-clips that were recently used in the "past". Considering the following sequence from Figure 4.9, it can be seen that sub-clip number 9 is repeated shortly after its first occurrence. Thus, a further constraint 4.8 is now added to the sequencing rules:

$$j \in \{j|P_{ij} > p_0\} \cap j \ni (l-r, l+r) \tag{4.7}$$

P defines the transition probability matrix, p_0 is a threshold probability.

$$j \ni Seq(n-v) \qquad v = 1, 2, \cdots, N \tag{4.8}$$

Seq represents the produced sequence, *n* is a counter index and *v* is a user defined value, which determines up to how recently used values are checked.

There is also the idea of rescaling the probability matrix (decrease the transition probability of sub-clips following after a disturbing sub-clip) when disturbing elements are noticed. The rescaling of the matrix is possible, but the difficulty is, that segregating the disturbing element when listening to resynthesized version is almost impossible. Within this work this suggestion could only be executed when very short (3-7 seconds) input signal were used, but this process involved the listening of every single sub-clip and the careful "reading" of the spectrogram together with the sub-clip borders of the original signal.

Another idea is to disable a disturbing sub-clip for a certain time (actually a certain number of sub-clips). Disabling means that a counter is established, which checks how many events are different from the disturbing clip that already passed. If a certain value (number of sub-clips passed that do not comprise the disturbing sub-clip) is exceeded, the disturbing sub-clip can be used in the output sequence. Once more, this is a solution that is possible for short-time input textures only.

4.6 New audio texture parameters

Summing up, I decided to use the following parameters so that the output texture of the audio texture algorithm are significantly improved:

- Sampling rate: 44.1 kHz
- Frame size/FFT-size: 512, 1024, 2048 and 4096 samples
- Hop size: Frame size/2
- MFCCs: 16, without the 0th coefficient
- Kernel: Kernelsize $\geq 13 \times 13$
- Delta for the peak detection: 2-9 (dependent on the frame size)
- Threshold for the peak detection: 0.00-0.2

4.7 Improving the analysis data: Natural grains

4.7.1 Using the parameters proposed by the authors

According to the papers of [Hoskinson and Pai, 2001; Hoskinson, 2002] the following parameters are used by the authors:

- Sampling rate: 44.1 kHz
- Frame size: 1024 samples
- Hop size: 256 samples (1/4 of the frame size)
- Wavelet decomposition level: 6
- Euclidean distance measure over 4 frames
- Mother wavelet: not specific, various wavelets are mentioned

Once more, in a first testing step the analysis parameters suggested by the authors were used. As I already mentioned above, I noticed that the sound examples from the sound database worked well also with the natural grain algorithm as longs as no textures were used that are made up of several streams. In a similar empirical approach I tried to decompose the parameters of the natural grain algorithm in order to improve the segmentation and the resynthesis of the sound textures.

4.7.2 Wavelet analysis

Since the Wavelet decomposition is performed frame wise, the same frame sizes are used as proposed for the audio texture algorithm above. As opposed to Section 4.7.1 the use of a bigger hop size (1/2 of the frame size) is suggested in order to have the same base of operations as the audio texture algorithm.

4.7.3 Choosing a mother wavelet

In the Java applet of [Hoskinson and Pai, 2001] miscellaneous wavelet functions such as Daubechie, Haar, Spline, Coiflet etc. can be selected. However, there are no general considerations available from [Hoskinson and Pai, 2001] about which mother wavelet should be used.[7] [Twicken, 1998] emphasizes

[7] Various wavelet filters can be chosen in the corresponding Java applet.

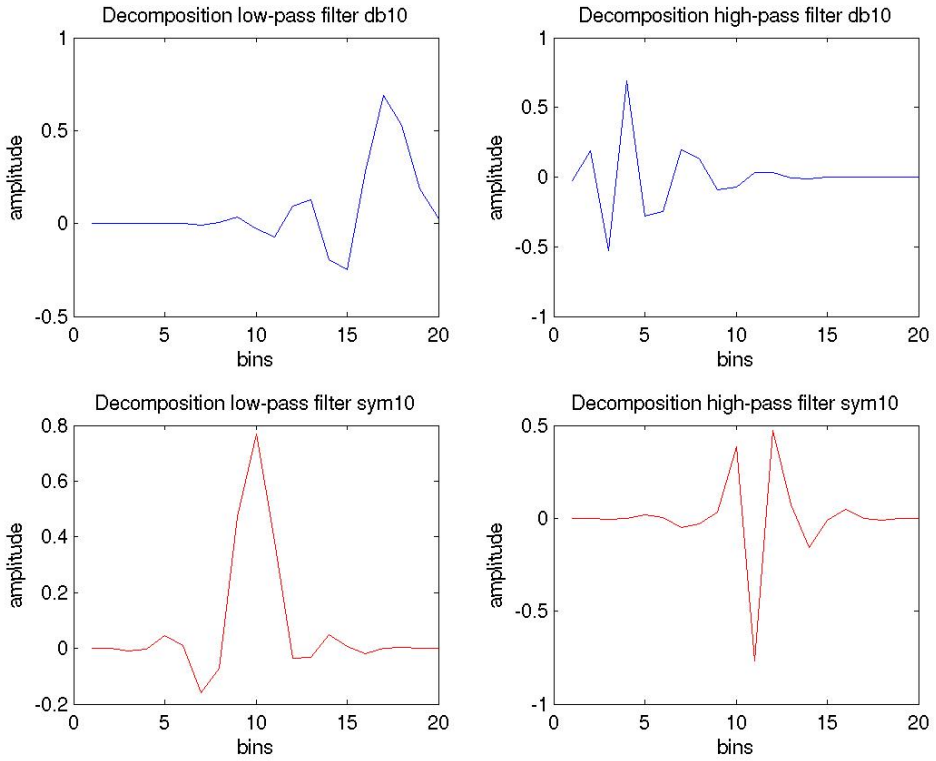

Figure 4.10: Daubechie 10 wavelet (upper two plots) and Symlet 10 wavelet (lower plots)

that a wavelet function should closely match the signal to be processed. This statement raises the question, which wavelet function matches best the input textures. There is no general wavelet function fulfilling the needs for all different sound textures. Even if a wavelet function was found that resembles a single water drop, there are so many different drops within an example of a rainfall that no wavelet function can be found, which fully covers that task. First, a Daubechie 10 wavelet was used (also in [Bar-Joseph et al., 1999]), which provided convincing results concerning the energy levels (see Figure 4.10). In a further step the natural grain algorithm was compared with the audio texture approach using beat signals (see Section 4.10 for more details). Using the Daubechie wavelet, which is asymmetric, always produced a small systematic offset when the local peaks in the novelty score generated by MFCCs and wavelets were compared. Due to this observation, I decided to use a Symmlet 10 wavelet (see Figure 4.10), which is the least asymmetric wavelet within the wavelet types that have compact support.[8]

4.7.4 Increase number of decomposition levels

In [Hoskinson and Pai, 2001] it is suggested to use a 6 level wavelet decomposition. When working with a sampling rate of 44.1kHz this leads to the following frequency bands:

1. 11025 - 22050 [Hz]
2. 5512.5 - 11025 [Hz]
3. 2765.25 - 5512.5 [Hz]
4. 1378.13 - 2765.25 [Hz]

[8] A function has compact support if it is zero outside of a compact set [Todd, 2006].

5. 689 - 1378.13 [Hz]
6. 0 - 689 [Hz]

The 6 level wavelet decomposition is directly adopted from a work in speech recognition [Alani and Deriche, 1999]. In their paper the authors argue that using a sampling rate of 16kHz, a 6 level wavelet decomposition provides a useful representation of speech. Sound textures do not have the characteristics of speech signals and spectral analysis shows that several sound textures bear information in the frequency domain below 689 Hz.

The Euclidean distance function represents the similarity between successive frames. Basically, the similarity measure is defined by the frequency changes in each Wavelet decomposition level. Thus, the higher the frequency resolution, accordingly the more Wavelet decomposition levels, the better segments can be detected in the Euclidean distance function. Therefore, it is necessary to perform a wavelet decomposition using more than 6 levels. A 9 level decomposition[9] seems much more appropriate for theses signals:

1. 11025 - 22050 [Hz]
2. 5512.5 - 11025 [Hz]
3. 2765.25 - 5512.5 [Hz]
4. 1378.13 - 2765.25 [Hz]
5. 689 - 1378.13 [Hz]
6. 344 - 689 [Hz]
7. 172 - 344 [Hz]
8. 86 - 172 [Hz]
9. 43,5 - 86 [Hz]

4.7.5 Euclidean distance function over more frames

The novelty score, described in Section 3.4.3, is obtained correlating the similarity matrix of the input texture with a two-dimensional kernel so that the score corresponds to frequency changes of the input signal. However, the quadratic Euclidean distance measure, which detects the similarity between frames is used over four frames only. It can be seen that there is a visual correspondence between the Euclidean distance function and the spectral changes of an input texture, but the distance function does not seem to be very significant. Thus, I modified and abstracted the parameters of the quadratic Euclidean distance function over four frames so that a weighting over more frames can be performed.

$$D(fb, fc) = \frac{1}{n} \sum_{i=1}^{n/2} \sum_{j=n/2}^{n} \sum_{k=1}^{m} (X_{i,k} - X_{j,k})^2 \tag{4.9}$$

The variable *n*, which has to be an even number, represents the number of frames that are weighted and *m* denotes the decomposition scales of a frame. Consequently it can be argued that the higher *m* and *n* the more the distance function corresponds to the frequency changes of the input signal and the more accurately the sub-clips are determined. Since the new distance function is smoother, the detection of the local troughs becomes easier and longer sub-clips are detected. In Figure 4.11 a Euclidean distance function over 4 and over 12 frames respectively can be seen. Weighting over 12 frames makes the function smoother and the detection of events on a larger time scale becomes easier.

[9] The DWT consists of log2(N) stages at most. *N* represents the frame size: log2(512)=9 stages, log2(1024)=10 stages, log2(2048)=11 stages.

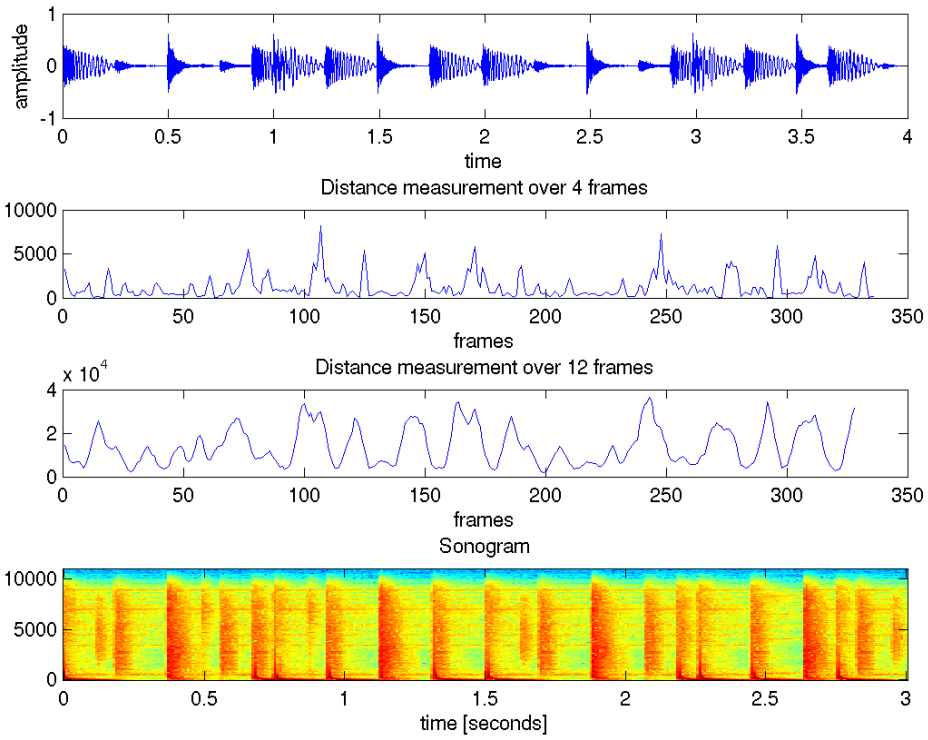

Figure 4.11: The Euclidean distance function computed over four (Plot 2) and twelve (Plot 3) frames respectively.

4.7.6 Getting the sub-clip borders

The local minimum detection is performed as described in Chapter 3. Merely the variable *minframe* is introduced as a user defined parameter to control the minimum of frames in a new sub-clip. The conversion from frames corresponding to local troughs in the Euclidean distance function to samples in the input texture is performed as described in Section 4.4.3.

4.8 Improving the resynthesis: Natural grains

4.8.1 Extending the markov chain

In [Hoskinson and Pai, 2001] the transition probability is computed from the Euclidean distance measure over the last two frames of any sub-clip and the first two frames of any other sub-clip. The common quadratic Euclidean distance between two vectors is defined as:

$$d(x, y) = \sum_{i=1}^{n} (x_i - y_i)^2 \tag{4.10}$$

If vector *x* equals vector *y*, the distance *d* would be zero. However, when the distance function for all the sub-clips is computed, the diagonal (distance of a sub-clip withitself) never gets zero, because the distance function is computed over four frames (the first two frames and the last two frames in a sub-clip). Consequently, there is a non-zero probability that sub-clip *n* follows itself. In a markov model the transition probability $P_{i,j}$, the probability from state *i* to state *j*, is given from every possible combination of *i* and *j*, including *i=j*, (see Figure 4.12). Thus, if a first order markov chain is used, repetitions of the

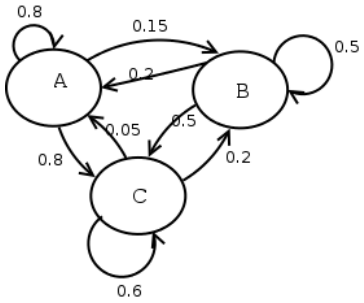

Figure 4.12: A simple markov model. The transition probability $P_{i,j}$ is given for every possible combination of i and j (including i=j).

same sub-clips are possible, because the transition probability is defined for that case. The *audio texture* algorithm and the extensions from Section 4.5.1 already provide various constraints for generating a sub-clip sequence that avoid repetitive patterns. In principle, these sequencing rules can be regarded as a markov model of a higher order. Therefore, I suggest to use the sequencing rules described in Section 4.5.1 also for the natural grain algorithm.

4.9 New natural grain parameters

Summing up, I decided to use following parameters:

- Sampling rate: 44.1 kHz
- Frame size: 512, 1024, 2048 and 4096 samples
- Hop size: Frame size/2
- Wavelet decomposition levels: 9
- Wavelet type: Symmlet 10
- Euclidean distance function over 12 frames
- Minimum frames of a sub-clip (*minframes*): 2-6 (frame size dependent)

4.10 Exchange features and methods

The introduced algorithms are very similar although they use different concepts for segmenting the original signal and different feature vectors. This fact gave me the idea to test whether the feature vectors could be exchanged and whether the principle ideas (finding the points where the signal changes dramatically versus detecting local troughs in the signal) of these methods can be used in both algorithms.

Actually, a comparison of these features (MFCCs and Wavelets) can only be approximated because the feature units are totally different. The energy of the detailled coefficients in each scale is represented in percent relative to the overall frequency of a frame whereas the MFCCs are cepstral coefficients, which reflect the energy balance in a frequency band.
However, the crucial difference between these methods is their different temporal synchronysation point of analysis. The MFCC computation is based on the linear Fast Fourier Transform (FFT). The point of analysis is always at the same time and due to the effect of windowing the absolute maximum in a frame is always in the center of a frame [Poli et al., 1991].
The wavelet coefficients on the other side are scaled and translated in every decomposition stage so that there is no temporal point of synchronization.

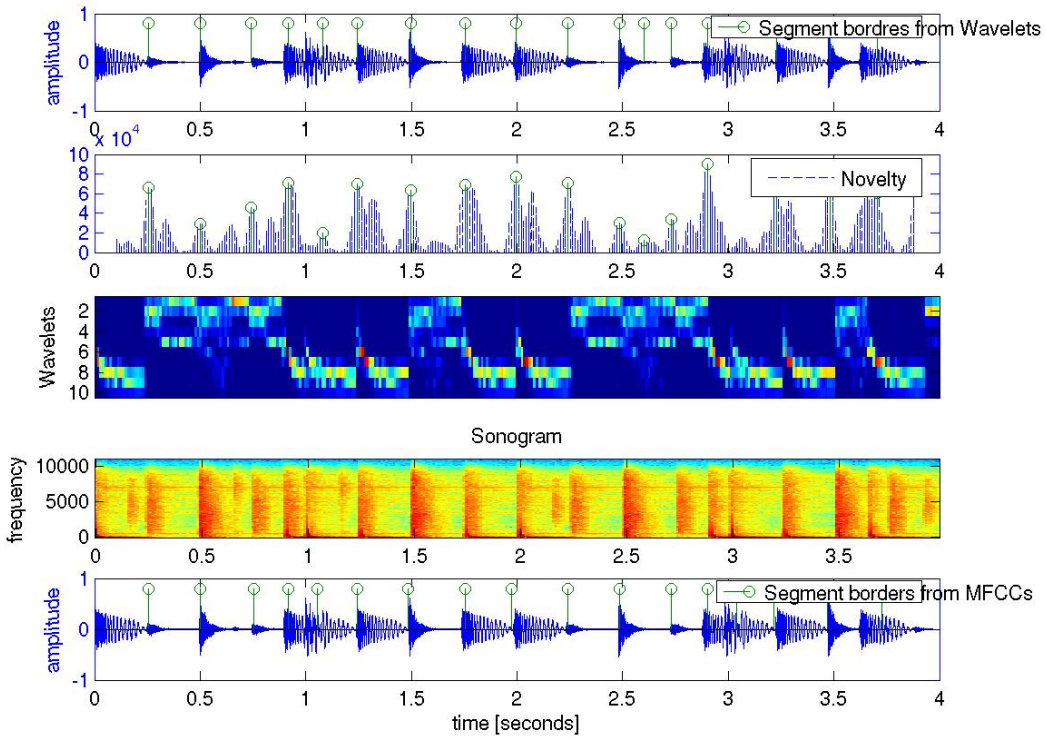

Figure 4.13: Most of the peaks are accurate to a sample block. The novelty score, depicted in plot 2, was built using wavelet coefficients. Plot 1 shows the original signal with the sub-clip borders from the wavelet analysis whereas the last plot shows the borders that are derived from the MFCC analysis.

4.10.1 Building the novelty score using wavelets

Since the generation of the similarity computation and the peak detection in the novelty score is originally developed for rhythmic audio signals by [Foote, 2000; Foote and Uchihashi, 2001], it seems obvious to use simple rhythmic signals for testing. The feature detection methods can be exchanged and although the methods are different, it is possible to detect mainly the same peaks accurate to an analysis frame. Basically, I found out that the result could be significantly improved when a Symmlet wavelet is used instead of the asymmetric Daubechie wavelet. In Figure 4.13 the sub-clip borders that result from the novelty score using either MFCCS or Wavelets can be seen.

Sound textures do not have a rhythmic structure. However, using input textures with distinct events shows that using Wavelets produces similar results (see Figure 4.14). The segment boundaries are not accurate to an analysis frame anymore because frequency and amplitude changes do not emerge that abruptly.

4.10.2 Detecting local troughs in the novelty score

The audio texture algorithm can also be changed in such a way that local minima are detected in the novelty score. This possibility is working but cannot be compared with the results of the natural grain algorithm. The novelty score is the result of a frame similarity measurement, which is correlated with a two-dimensional kernel in order to detect regions of changes in the signal.

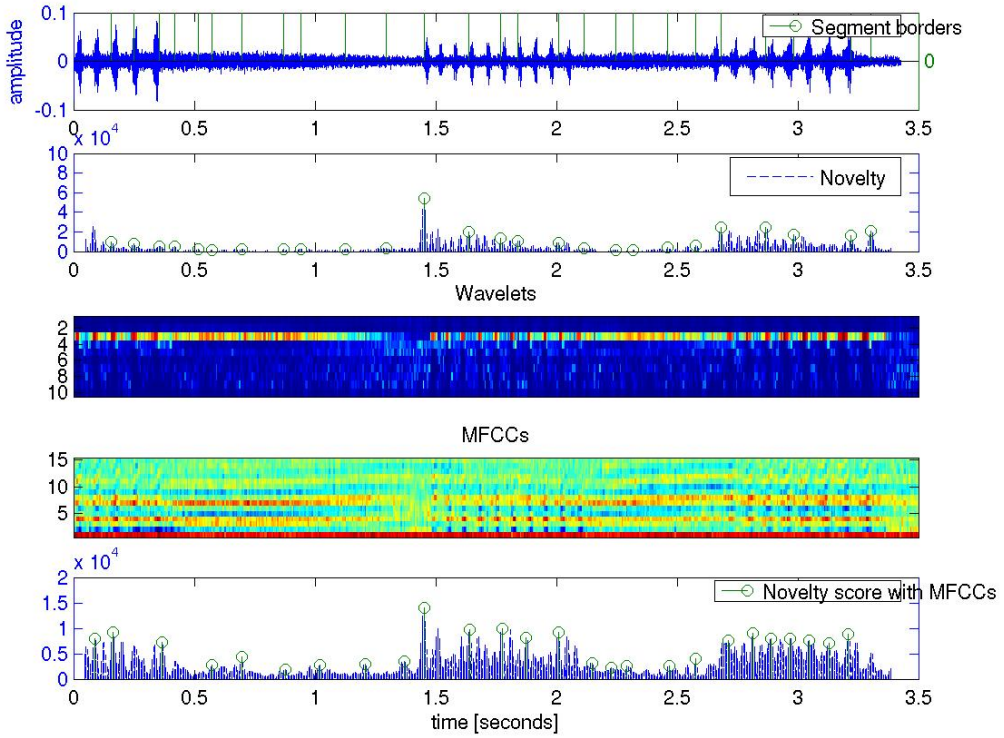

Figure 4.14: Chirping birds: Peaks are not detected in the same frames of the novelty score anymore but still produce similar results because the frequency changes in the signal are smoother.

4.10.3 Building the Euclidean distance function using MFCCs

The natural grain algorithm can also be built using MFCCs. However, due to different units and the specialty of the sub-clip generation (a local minimum has to lie beneath a certain threshhold and a minimum is only selected if it is at least *n* frames apart from the previous one), the sub-clipborders are not exactly the same.

4.10.4 Finding local peaks in the Euclidean distance function

The whole concept of the natural grain algorithms can be reversed of course so that local peaks are detected. However, as explained above, the distance function cannot be used as a measure of novelty and therefore a comparison with the audio texture approach does not make sense. Reversing the concept would also contradict with the idea of *a natural grain*, which refers to points in the signal ”where some kind of events start or end that cannot be broken up further”.

4.11 Conclusion

In this chapter I demonstrated that the audio texture and the natural grain were successfully modified for the segregation of perceptually perfect segments of input textures. Furthermore, the resynthesis could be successfully improved so that no repetitions can be heard and so the output texture seems natural.

Although, focusing on the parametric improvements of both algorithms, the audio texture algorithm turned out to be the preferred method.

- The novelty score more precisely represents the spectral changes of the input texture since it originates from the two-dimensional correlation of the similarity matrix and the Hamming-kernel.
- The transition probability is derived from similarity measures between any two entire sub-clips (and not only between the first two and the last two frames between any two sub-clips as proposed by the natural grain algorithm). This is very important, since sub-clips might get very long, due to the usage of a threshold for the peak detection in the novelty score.
- Listening to single sub-clips showed, that the audio texture approach more likely produces segments that correspond with our auditory segregation.

Chapter 5

Real-time Sound Texture Generator

" C'est le rhythm qui fait la texture "

[Pour une rhétorique de l'image, Seuil 1992]

The main idea of the Sound Texture Generator (STG) is to facilitate an easy generation of sound textures without getting into coding details. Moreover, the real-time generator, as opposed to the offline Matlab implementation, is not CPU expensive, so that sounds can easily be tested and saved for further applications.

As a proof of my concept, as presented in Chapter 4, a high-level Pure Data interface is provided demonstrating both presented algorithms and incorporating the parametric improvements. The graphical interface is intended as an application for demonstration purposes. Furthermore, several parameters and functions for controlling the concatenation of the segments and as well for testing the sequence patterns are provided.

5.1 Real-time software Pure Data

The demonstration prototype is built using the graphical real-time graphical programming environment Pure Data (PD).
PD is a real-time graphical programming environment for audio, video, and graphical processing, which is available for all platforms.[1]

PD represents a major branch of the family of patcher programming languages known as Max originally developed by Miller Puckette and his team at IRCAM. PD is an example of a dataflow programming language. In such a language, functions or "objects" are linked or "patched" together in a graphical environment modeling the flow of the control and audio. In this chapter several technical terms are used, which might be new to a reader who is not familiar with PD. For a more in-depth introduction to the PD environment and its programming objects the reader is referred to [Puckette, 2006].

PD was selected as a building software for the sound texture protype because, in contrast to the numerical computing Matlab environment, it is designed for real-time processing and allows for a fast and interactive modification of the parameters and methods.

5.1.1 Software requirements of the STG

In order to run the STG the following conditions are required:

[1] `www.puredata.org` (accessed Nov. 17. 2006)

- PD version 0.40
- zexy[2]
- iemlib[3]
- iemmatrix[4]
- shell and stripdir[5]
- Matlab 7.0

5.2 Two-level system of the algorithmic structure

On the basis of the available Matlab analysis functions, I decided to construct the STG as a two-level system. This means that the implementation of the STG prototype incorporates an offline analysis (Level 1, performed in Matlab) and the real-time resynthesis (Level 2, implemented in PD) (see Figure 5.1). However, both levels can be controlled from the interface (see Section 5.2.1 and 5.3).

Level 1

The first level comprises the following steps:

- Feature extraction of the input texture (either MFCC or Wavelet)
- Similarity measures
- Creation of the sub-clips (due to the novelty score or the Euclidean distance function)
- Transition probability of sub-clips

From Level 1 the borders of the sub-clips and the transition probability of the sub-clips are passed to Level 2 (see Section 5.2.1 for more details).

Level 2

In the second level the following steps are performed;

- Sequencing based on the transition probability
- Concatenative synthesis of the sub-clips

5.2.1 Analysis: Level 1

Since the analysis data (sub-clip borders and transition probability of the sub-clips) from Level 1 is passed to Level 2, the data must be saved in such a way that PD can interpret it. For reading the Matlab data the PD *iemmatrix* library is used, which is able to deal with matrices saved as textfiles that are created by an external program. Therefore, the borders for the sub-clips and the transition probabilities are saved in Matlab as textfiles with the suffix **.mtx* so that the iemmatrix objects can access these files.

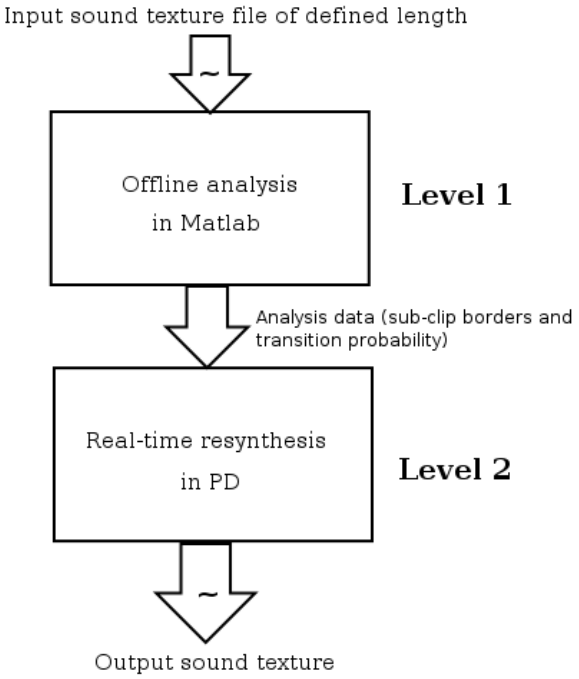

Figure 5.1: Two level implementation structure for building the real-time STG.

```
#matrix n 2
a11 a12
a21 a22
.
.
an1 an2
```

Listing 5.1: Example matrix for the sub-clip borders

```
1 #!/bin/sh
2
3 echo "starting matlab from  PD executing audiotexture analysis"
4
5 ./matlab -nosplash -nodesktop -r "audiotexturePD(${filename},${fft_size},${delta},
      ${threshold),quit"
6
7 echo "bang;"| pdsend 6666 localhost udp
```

Listing 5.2: Parts of the shell script: matlab1_start.sh.

Consequently, a matrix that is read by PD, such as for example the matrix representing the n sub-clip borders of an input texture, looks like Listing 5.1.

The first entry, starting with a hash character, defines a matrix with n rows and two columns (the first row defines the start point of a sub-clip, the second row the end point). The entries for the values of the matrix start in the second line of the textfile.

Starting Matlab from PD

Already in the introduction of this chapter I mentioned that the generation of sound textures with the STG should be simple and coding details should be avoided. Now that the implementation structure of the STG incorporates an offline analysis with Matlab, I decided to develop a script that enables the starting of Matlab from the PD interface. Thus, Matlab is controlled from PD and no programing code has to be checked. Only the control parameters for the analysis have to be inserted in the PD function call. With this solution I provided an easy access to the offline analysis from PD that enables also users having no programming knowledge to use the STG.

In the *analysis* abstraction (see Section 5.3) the analysis data (sub-clipborders and transition probability matrix) for the audio texture and the natural grain algorithm can be computed. Basically, the prototype already includes the analysis data for several example textures. Thus, the idea of the *analysis* abstraction is to use it on the one hand when a new soundfile is loaded and on the other hand when the user is not satisfied with the quality of the output textures. Two shell scripts (matlab1_start.sh(A) and matlab2_start.sh(NG))[6] have been developed that can be executed via the *shell* object[7] (see Figure 5.2). The shell scripts need parameter values, such as frame size, decomposition levels etc. in order to run properly. These parameters can be directly inserted as arguments in the shell script call within the PD patch. To accelerate the program compilation, Matlab is started without a GUI (see Listing 5.2) and the program is closed after the function call. If the execution of the Matlab function is finished, PD receives a *bang* message via a user datagram protocol (udp) socket. Hence, the new analysis data can be used for the sound texture generation.

5.2.2 Resynthesis: Level 2

As soon as the analysis data is available as an input from Level 1 the whole resynthesis can be performed in real-time corresponding to the algorithmic approach introduced in Chapter 4. The creation of the

[2] from pd-cvs/externals/iem/zexy/
[3] from pd-cvs/externals/iem/iemlib/
[4] from pd-cvs/externals/iem/iemmatrix/
[5] from pd-cvs/externals/ggee/control/
[6] In the patch audio texture is abbreviated with the letter *A* and natural grain with *NG* respectively.
[7] The shell object in PD enables the connection of another program. This object needs the name of a shell script as argument.

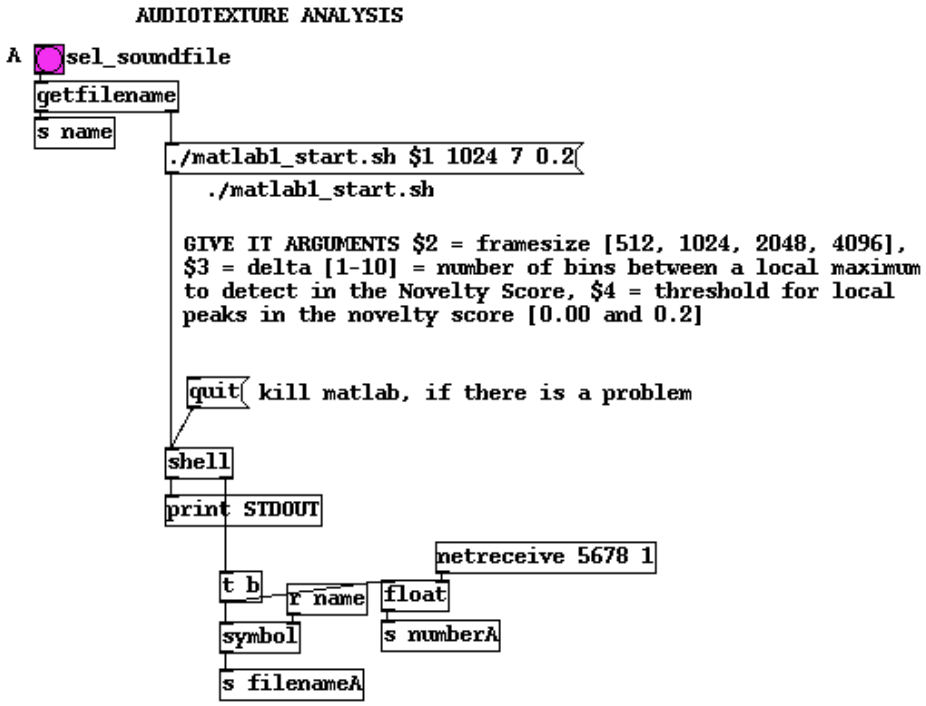

Figure 5.2: Screenshot of the *analysis* abstraction. The parameter values for the shell scripts are user defined. If no values are provided default values are used.

sequence patterns based on the transition probability is implemented in the patch named Probability.pd[8]. Since the implementation of this patch corresponds with the sequencing rules presented in Section 3.4.5, no more details are given here.

Concatenative synthesis of segments

While testing the algorithms in Matlab the crossfading of the sub-clips could be implemented accurately to a sample. The algorithmic approach was simple because in an offline computation the algorithm does not need to wait until a clip is played and every procedure is executed sequentially.
In PD on the other hand the solution is different because every selected sub-clip can be heard immediately. Hence, in the *player* abstraction two *tabplay~* objects[9] are used which alternate (see Figure 5.3). Whenever a sub-clip is selected the length of the sub-clip in milliseconds is computed. While the sub-clip is played by the first *tabplay~* object a delay object counts the time of the current sound file length minus the fade time. When this time is reached the first sub-clip is faded out. At the same time a new sub-clip is triggered, the other *tabplay~* object starts and the sub-clip is faded in. This procedure continues until the process is switched off by the user.

5.2.3 Informed versus uninformed synthesis

Very fine grained, stochastic sound textures, such as fire, rain and applause obviously work well with every algorithm, and parameter modifications do not seem to have a great influence on the acoustic result. Therefore, it is doubtable whether the algorithmic analytical process is necessary for these types

[8]The patch is located in the abs-folder (see Appendix A.2).
[9]A tabplay object plays part of a soundfile taking the begin and the end point of a subclip as input arguments.

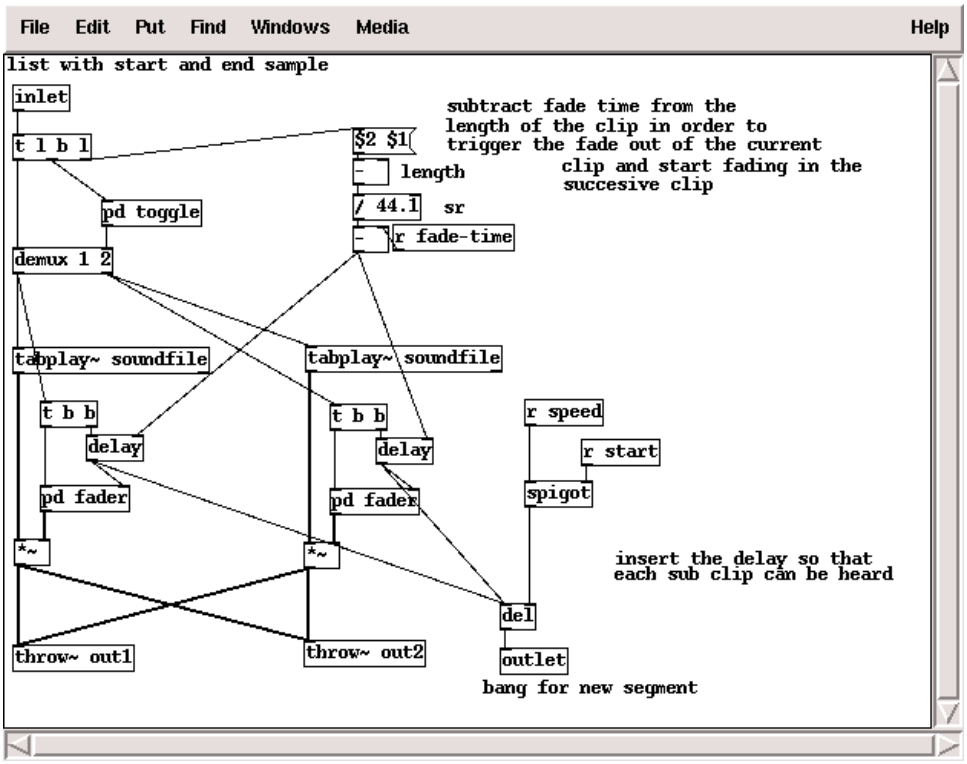

Figure 5.3: Screenshot of the *player .pd* abstraction where sub-clips are concatenated.

of signals. Since this assumption can be easily tested in PD, two abstractions are created. Both can be started from the main patch in the *select method* unit (see Section 5.3).

Random sequence order

In the first abstraction the sub-clip borders computed in Level 1 are used, but all the sequencing rules based on the transition probability from Section 4.5.1 are neglected. The sequence is only determined by a pseudo random number generator that emits random values in the range of the available sub-clips.[10] It was observed that as long as the random numbers are uniformly distributed no difference can be noticed in comparison with an algorithmically determined sequence. However, there is no control that value x must not follow itself and also neighboring sub-clips can be selected which often produces disturbing repetitive patterns that can be heard. The random sequence order was also tested with signals that have noticeable forground events, for example environmental sounds with significant bird chirps. With these signals the sound quality is significantly lower. Repetitions can be heard owing to the uncontrolled random generator. Furthermore, the same sub-clips are repeated one after the other so that the resynthesized version is not considered to be natural anymore.

Random texture synthesis

The second abstraction ignores the analysis data from level 1. Thus, this approach generates random sub-clips borders. According to a user defined average clip length[11], ranging between 13 and 130 milliseconds, a random sub-clip border matrix is created. Again the pseudo random number generator is used to define the sequence order. Now the results are very dependent on the average clip length. Below

[10] Actually this number is defined by the size of the sub-clip border matrix, which has n rows and 2 columns.

[11] Always some jitter is added to the average length to simulate an altered sub-clip length. The corresponding patch is called randomClipBorders.pd.

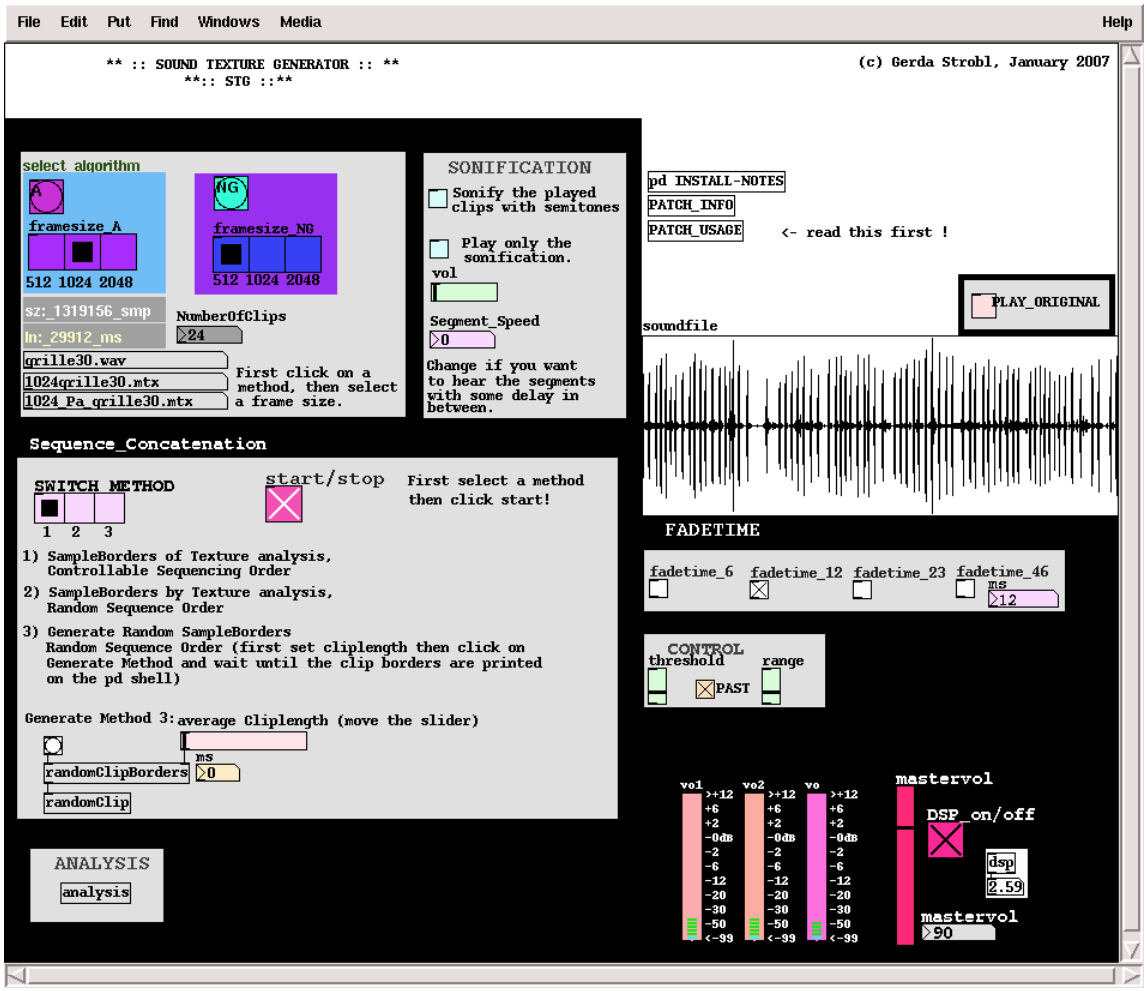

Figure 5.4: Screenshot of *main.pd*, the sound texture control interface. All the functions can be controlled from the main patch.

50 milliseconds the resynthesized output textures versions do not convince for any input texture material. Using a longer average clip length improves the sound quality but still, due to the randomness of the sub-clip borders forground events get cut somewhere in between, which proves that for these kind of signals the informed synthesis produces output textures that are to a great deal better than the results of the random texture synthesis.

5.3 Interface structure

The whole interface can be controlled from the Graphical User Interface (GUI) in the file *main.pd* (see Figure 5.4).

The control GUI contains the following graphical units:

- *select algorithm*: Either the audio texture or the natural grain algorithm can be selected. After the selection of an algorithm a dialog box pops up where a soundfile has to be loaded. In a next step also the frame size has to be selected. In order to access all the matrix data (sub-clip borders and transition probability) the name of the sound file (e.g. fire.wav) is extracted as a string without the ".wav" extension so that the matrix suffices (*.mtx) for corresponding matrices are created (e.g. 1024fireP_AT.mtx) and so the data is automatically loaded.

In this unit the analysis is not computed anymore and the user only selects the data that was generated in Level 1 before.

- *switch method (sequence concatenation)*: Three methods can be selected. The first method starts the analysis/resynthesis approach introduced in this thesis using the parameters calculated in the offline analysis. The second method again uses all the analysis data, but performs a random sequence concatenation. Finally, the third method permits the real-time generation of random length clips and random sequencing.
  The second and the third method are fully explained in Section 5.2.3 above.

- *fade time*: According to a chosen frame size a corresponding fade time in milliseconds, that is half the frame size, is set (512: 6ms, 1024: 12ms, 2048: 23ms and 4096: 46ms). The use of a higher fade time is suggested when working on a larger frame size in order to get smoother cross fades. As a matter of course, every fade time can be used with any frame size.

- *control*: The range factor r and the threshold P_0 described in Section 3.4.5 can be modified with the respective slider. Pushing the button[12] *past* leads to a comparison of the current selected clip number with previous clip numbers. If the button is deactivated all the values are allowed to pass even though they were just selected. The underlying idea of this function is fully explained in Section 4.5.1.

- *play original*: The original sound file can be played.

- *audio control functions*: This unit contains the general audio control functions, such as master volume, vu-metres and Digital Signal Processor (DSP).

- *analysis*: Opening the *analysis abstraction* enables the user to start the offline Matlab analysis from a PD patch (see Section 5.2.1 for more details).

- *sonification*: If the first method in the sequence concatenation unit is selected the index of each sub-clip can be sonified with a sine tone. The frequency specifies the index number in semitones. Changing the values of the *Segment Speed*, which correspond to milliseconds, inserts a delay between successive sub-clips so that each sub-clip can be heard alone if a long delay is selected (see Section 5.4 for more details).

5.4 Repetition control using sonification

Since it is difficult to control the sub-clip sequence order of sound textures which are longer in duration, an easy way of checking the appearance of repetitive patterns is to use a sonification of every sub-clip. In the current implementation sub-clips are consecutively numbered and so the index numbers correspond to semitones.

With the sonification it can be demonstrated that owing to the strict sequencing rules no repetitive patterns can be heard. Furthermore, it can also be proved acoustically that the *past* function perfectly avoids sub-clips that appear too often successively because as soon as the *past* button is deactivated undesired semitones can be heard in the sonification.

Generally, I noticed that the longer the duration of the input sound texture the better the quality and the vivacity of the output texture. The best results were achieved with input texture lengths longer than 25 seconds. This observation seems logic but has to be mentioned in that context since common input sound textures lengths in the literature are not longer than maximally 15 seconds.

[12] In PD the ”button” is called toggle.

5.5 Constraints of the prototype

As I explained above in Section 5.1, the signal analysis is performed offline. Even though the analysis can be started from PD, this is not the most elegant solution. So far the prototype has run on a Linux[13] system only, but might be easily ported to other platforms. As the installation of all the externals can be difficult, I considered to intergrate the whole patch on a Linux Live-CD. Unfortunately this cannot be done since Matlab is not an open-source program. I also tried using octave[14] instead, but the MFCC computation is extremely slow and unfortunately the wavelet toolbox commands from Matlab are not integrated into octave yet.

Changing the threshold and range parameters for the sequence generation sometimes produces a stack overflow error, which freezes the program. Since this is not intended, a loop controller is implemented which stops the program as soon as the loop index exceeds a certain threshold. Due to that a warning message is printed on the PD console. Thus, the sequence control parameters have to be changed and the whole process has to be reset with the start button.

Pressing the start button always resets the sequencing control parameters. If this button is not activated when different frame sizes are tested one after the other it can happen that the control parameters do not fit with the new transition probability matrix anymore so that the process also gets stuck. Therefore, it is suggested to always start and stop a process before going on.

[13] In this case the prototype was built on a debian system.
`http://www.debian.org` (accessed Nov. 18. 2006)

[14] `www.octave.org` (accessed Nov. 2. 2006)

Chapter 6

Reflections upon sound textures

" Whatever may come, the world keeps revolving,... that is all just a little bit of history repeating. "

[Propellerheads, History Repeating 1998]

Right at the beginning of this work, before investigating the details of the algorithmic parameters, it could be heard, that very stochastic, fine grained sound textures, such as fire and liquid sounds, work perfectly with both algorithms. This could be simply proved by listening to every single sub-clip[1] in the real-time prototype (compare Section 5.3). Due to the stochastic structure every segment boundary seemed to be valid, and as stated in Section 5.2.3, it was even hard to notice any difference when comparing the informed synthesis (each segment is statistically dependent on its predecessor) with the random sub-clip selection. These sound textures comprise mostly natural sounds and some machine sounds that are perceived not as single events, but as a continuous dense stream. Attributes for describing these sounds include: crackling, crumpling, droning, rustling and buzzing.

As soon as sounds, which contain several streams that can be easily segregated by a listener (compare Section 3.1) were used, the algorithms produced unsatisfying results. Moreover, the same listening observations that were also found in the literature (compare Section 2.2) could be noticed.
Fortunately, due to the parametric improvements presented in Chapter 4, the sound quality of the output textures could be significantly improved in such a way that also soundscapes can be used as input textures.

6.1 What is a perfect input texture?

Just as explained in Chapter 4 several parametric improvements were made so that the sound quality of textures consisting of several streams could be significantly improved and noticeable repetitions were removed. However, I discovered that there are still some recordings that work better than others.

A sound texture that works very well is the recording of chirping birds in a summer garden (see Figure 6.1). Without knowing the signal it can be seen that the bird chirps are very well distributed over the whole spectrum. Furthermore, when listening to that sound file, it can be heard that there is a very good "ratio" between foreground sound events and background sounds. Moreover, the background noise is very smooth and clear. Due to this special structure very long sub-clips are detected and when listening to the single sub-clips the segment borders make sense. An extract of the 26 second sound file and its segment borders is depicted in Figure 6.2. The resynthesized version just sounds natural, and

[1] Setting the segment speed to 1000 milliseconds, inserts a 1 second delay between every sub-clip so that a single sub-clip can be heard alone.

it is of particular importance that the events seem to be placed perfectly in time and the peaceful, slow impression of the soundfile is preserved.

However, choosing the threshold for the peak detection in the novelty score is a crucial step (compare Section 4.4.4). Even though the presented bird signal seems perfect, the segmentation is further improved using a threshold so that only significant streams are segregated and the rest of the segments contain the noisy background sound.

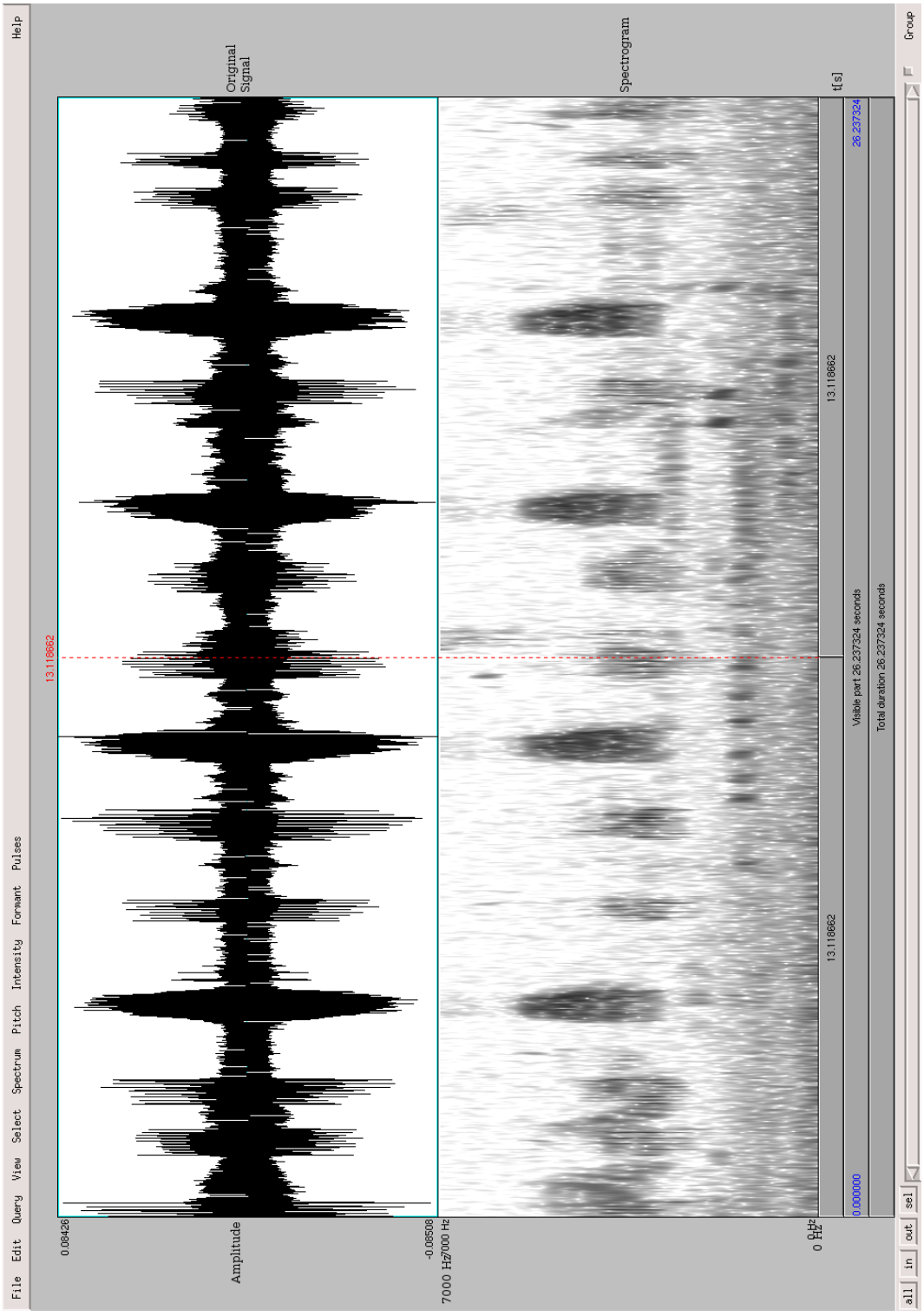

Figure 6.1: Screenshot of the spectrum and the original signal. Already from the spectrum it can be seen that the sound events are very well distributed over the whole spectrum and over different frequency ranges.

6.2 Imperfect sound textures

In search of perfect sound texture input signals the following types and characteristics produced nonsatisfying results:

- **Waves and wind:** A rolling wave lasts several seconds and every wave involves an increasing and decreasing envelope as can be seen in Figure 6.3. Wave signals were tested in the Matlab environment only. Even though large frame sizes were used, single waves were not captured as an entire stream. Hence, the waves were cut into pieces. Due to the changing envelope the resynthesized version always sounded like a noise signal which has constant blocks of different changing envelopes.
  The same applies analogously for ”howling wind” signals. The undulating amplitude envelope of these signals cannot be resynthesized. Therefore, the new sequences sound like a chopped concatenation of sequences.

  I still consider ocean waves and wind signals as sound textures. However, using the parameters proposed in this thesis, the algorithms do not capture the properties of these signals. I am sure that using very large analysis parameters might succeed but this was not tested in the context of my work.

- **Speech background with single words:** Speech background sound works well when a large frame size is used and as longs as single words are not intelligible.
  As soon as a single word can be retrieved, it still happens that the word is cut into two pieces. This might happen because the word contains a prominent spectral change which results in a local peak in the novelty score, thus, the word is split. Furthermore, as soon as a single word can be understood and remembered, the more often it reappears in the resynthesized version, the more it sounds like an illogical repetition.

  In my opinion it should really be questioned if human utterances are considered as textures. I think this question involves more than just asking: How many people have to talk so that we perceive a texture. I would also ask - how much do we have to comprehend that we consider speech as texture and is a single intelligible word comparable to a bird chirp that appears again and again?

- **Quasi-rhythmic textures:** Sounds such as applause, periodic engine noise et cetera are considered as sound textures in Section 1.2.1. Usually these sounds do not cross one’s mind when thinking about rhythmic sounds. But listening to theses signals over some time makes it evident that there is some rhythmic structure.

  Rhythm is not considered in the presented algorithmic resynthesis structure. Furthermore, segments cannot be concatenated in a rhythmical order. Therefore, I rather suggest to exclude quasi-rhythmical signals from the definition of sound textures.

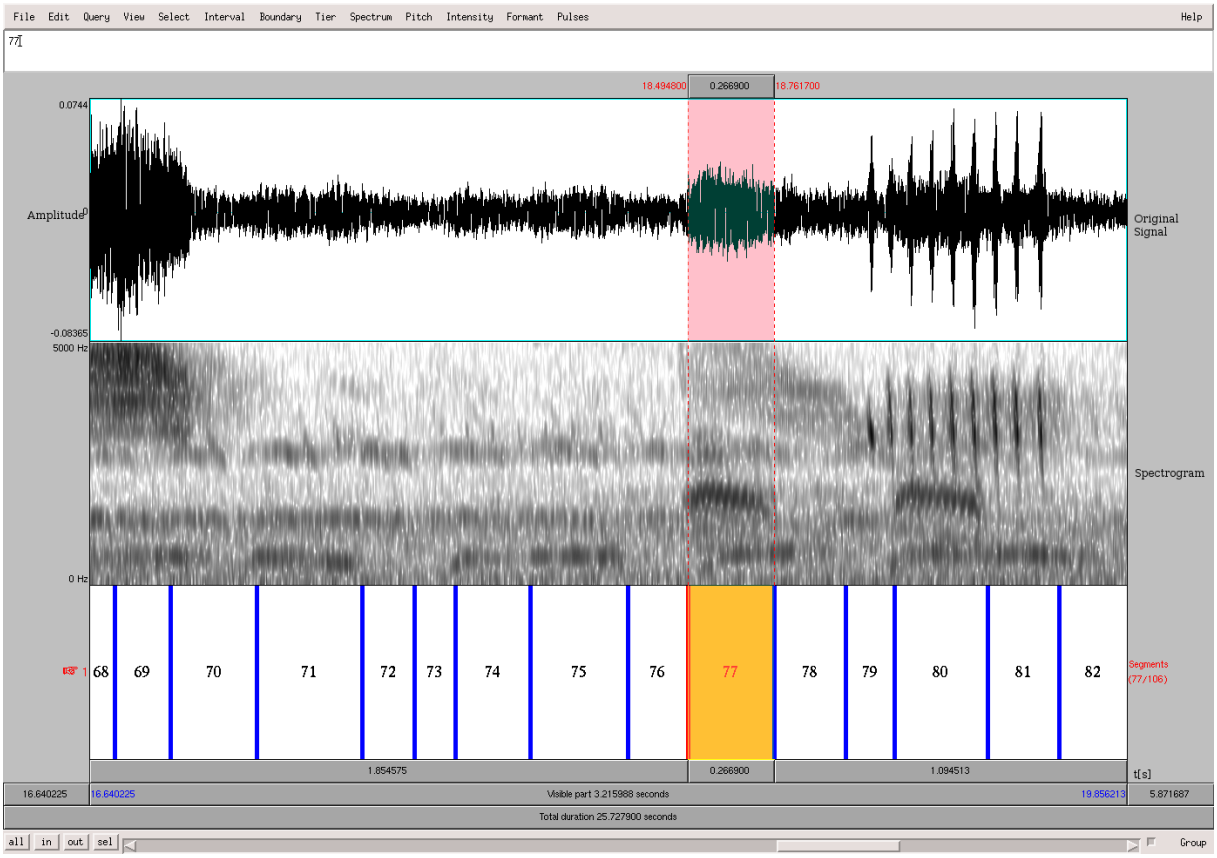

Figure 6.2: Due to the perfect distribution of events in different frequency ranges, the segment borders just follow frequency changes of the events.

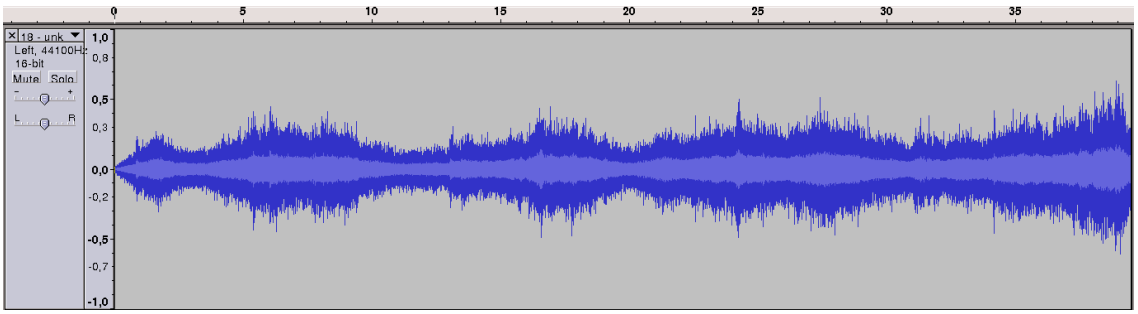

Figure 6.3: Wave signal: The undulated envelope of a single wave which lasts some seconds can already be seen in a simple audio editor. With the current parameters these long-term changes cannot be captured.

Chapter 7

Conclusion

A primary goal of my thesis was to give a general idea on the current state of the art of sound texture modeling in computer music. I tried to determine the scope of different fields of acoustic texture and starting from the literature research I investigated two algorithms (audio texture and natural grains) for creating a sound texture generator that creates output textures of undetermined length out of a short input texture. By an empirical approach I tested and modified these algorithms so that the quality of the output textures could be successfully improved. While modifying the parameters, I discovered that these algorithms can be used for segregating the input texture into perceptually meaningful units. Therefore, a major contribution is that due to the parametric changes, segments are only created at points that make sense to our auditory perception. As a proof for my investigations I created a real-time sound texture interface where both algorithms can be tested.

In the first chapter of this thesis I enumerated common sound texture examples, which were tested in the algorithmic environment. Owing to the different signal properties the finding of a general parametric analysis setup was not possible. However, with the parametric modifications of my work, I could particularly improve the construction of output textures consisting of significant foreground sounds that are rather associated as soundscapes or environmental sounds. Thus, after all the investigation on sound textures I still have to ask: What is a sound texture? In my opinion there is the need for a clear typology of sound textures. Precise constrictions have to be made so that the scope of these signals gets smaller which would simplify the analysis for these types of sounds. We can define speech and we can define music, therefore, also a distinct definition for sound textures should be constructed.

Apparently, there is the known issue that texture is a language-related term. In English it seems clear that the sound of burning wood, ”fire”, is considered as a sound texture, whereas denoting the same happing in German as ”Textur” might astonish a native German speaker. Hence, it would be interesting to find out if there is a linguistic equivalent in other languages.

It may also be intersting to have more fundamental resarch on perception based signal research that incorporates aesthetic aspects so that terms like sound quality or perceptually meaningful could be clearlier defined.

Since the generation of sound and music textures is very common in contemporary music composition, I would like to modify the real-time STG so that it can also be used as a composition tool.

As stated in the introduction, several 20th century composers have a very precise idea of what a texture is. Perhaps findig parallels between music and computer music research literature could enable a better understanding for these sound classes. Thus, one of the open points would be to find out if these disciplines are consistent in terms of texture.

Appendix A

Appendix

A.1 Mel scale

In 1937 the Mel scale was proposed by Stevens, Volkman and Newman. The Mel is a perceptual scale of pitches judged by listeners to be equal in distance from one another. The reference point between this scale and normal frequency measurement is defined by equating a 1000 Hz tone, 40 dB above the absolute threshold of hearing, with a pitch of 1000 Mel (see Figure A.1).

Above about 500 Hz, larger and larger intervals are judged by listeners to produce equal pitch increments. The name Mel comes from the word melody indicating that the scale is based on pitch comparison [Wikipedia, 2006b].

The frequency in Hertz can be converted to the Mel scale using the following equation :

$$m = 1127.01048 * log(1 + f/700) \tag{A.1}$$

And vice versa, from Mel to Hertz:

$$f = 700(e^{m/1127.01048} - 1) \tag{A.2}$$

A.2 CD-ROM

The accompanying CD-ROM contains the following items:

A.2.1 Matlab files

The folder Matlab contains the following files:

- audiotexture.m (Implementation of the improved audio textures algorithm) and audiotextureOld.m (original implementation without improvements)
- naturalwavgrain.m (Implementation of the improved natural grains algorithm) and naturalwavgrainOld.m (original implementation without improvements)

The following extra functions are required by the algorithms:

- get_wav.m (GUI for selecting a sound file)
- localPeak.m (local peak detection)
- lmin.m (local minimum detection)
- ma_mfcc.m (MFCC computation)

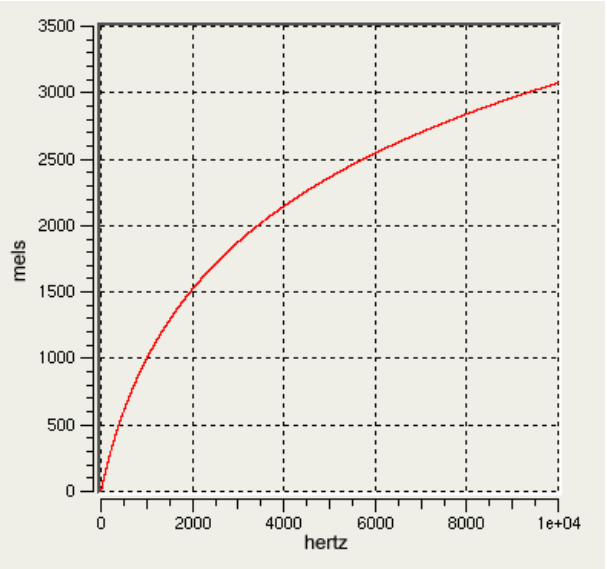

Figure A.1: Mel/Hertz Scale [Image extracted from [Wikipedia, 2006b] under the terms of the public domain copyright.]

A.2.2 Pure Data texture generator

The folder texturePD contains the following files and directories:

- start_pd.sh (startup shell script for the generator)
- main.pd
  The main patch is automatically started by the startup script See Section 5.3 for details.
- samples (sound files)
- abs (PD abstractions required by main.pd)
- at (*.mtx data of the audio texture algorithm)
- ng (*.mtx data of the natural grains algorithm)
- data (textfiles containing the analysis information about the created matrix data from the folders at and ng)
- matlabfiles (audiotexturePD.m and naturalPD.m)
- matlab1_start.sh (shell script for starting the audio texture Matlab algorithm from PD, compare Section 5.2.1)
- matlab2_start.sh (shell script for starting the natural grain Matlab algorithm from PD, compare Section 5.2.1)

A.2.3 praat

- mtx2TextGrid.awk (convert PD-mtx-Files to Praat-TextGrid-Files)
- TextGrid2mtx.awk (convert Praat-TextGrid-Files to PD-mtx-Files)

Bibliography

A. Alani and M. Deriche [1999]. *A Novel Approach To Speech Segmentation Using The Wavelet Transform*. In *Fifth International Pocessing and its Applications, ISSPA 99'*, pages 127–130. Brisbane, Australia. (Cited on pages 26, 28 and 43.)

M. Amadasun and R. King [1989]. *Textural features corresponding Textural Properties*. In *IEEE Transactions on Systems, Man and Cybernetics*, pages 1264–1274. (Cited on page 2.)

K. Andrews [2006]. *Writing a Thesis: Guidelines for Writing a Master's Thesis in Computer Science*. Graz University of Technology, Austria. `http://ftp.iicm.edu/pub/keith/thesis/`. Accessed on 1th Aug. 2006. (Cited on page ix.)

E. Asimov [2006]. *The indiscribable texture of wine*. The New York Times. (Cited on page 2.)

M. Athineos and D. Ellis [2003]. *Sound Texture Modelling with Linear Prediction in both Time and Frequency Domains*. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing ICASSP '03*, pages 648–51. (Cited on pages 4, 10, 11 and 14.)

J.-J. Aucouturier and F. Pachet [2004]. *Improving Timbre Similarity: How high's the sky?* Journal of Negative Results in Speech and Audio Sciences, 1. (Cited on pages 18 and 33.)

Z. Bar-Joseph, D. Lischinski, S. Dubnov M. Werman, and R. El-Yaniv [1999]. *Granular Synthesis of Sound Textures using Statistical Learning*. In *Proceedings of the International Computer Music Conference (ICMC)*, pages 178–181. Beijing. (Cited on pages 10 and 42.)

Z. Bar-Joseph, R. El-Yaniv, D. Lischinski, and M. Werman [2001]. *Texture Mixing and Texture Movie Synthesis Using Statistical Learning*. In *IEEE Trans. Visualization and Computer Graphics*, pages 120–125. (Cited on page 9.)

B. Behm and J.R. Parker [2004]. *Creating Audio Textures by Samples: Tiling and Stitching*. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing ICASSP '04*, pages 317–320. (Cited on pages 4, 10 and 14.)

N. Bernadini [2004]. *Sound Textures and Expressiveness*. Workshop on Sound Textures, Canazei. (Cited on page 4.)

A. Bregman [1990]. *Auditory Scene Analysis*. The MIT Press. (Cited on pages 9 and 15.)

F. Brosius [2002]. *SPSS11*. mitp Verlag, Bonn. (Cited on page 17.)

M. Cardle, S. Brooks, and P. Robinson [2003]. *Directed Sound Synthesis with Natural Grains*. Proceedings of the Cambridge Music Processing Colloquium 2003 (CMPC 2003). (Cited on page 4.)

M. Cooper, J. Foote, E. Pampalk, and G. Tzanetakis [2006]. *Visualization in Audio Based Music Information Retrieval*. Computer Music Journal, 30(2), pages 42–61. (Cited on page 21.)

Y. Dobashi, T. Yamamoto, and T. Nishita [2003]. *Real-time Rendering of Aerodynamic Sound using Sound Textures based on Computational Fluid Dynamics*. In *ACM Transaction on Graphics*, pages 732–740. (Cited on page 13.)

Y. Dobashi, T.Yamamoto, and T. Nishita [2004]. *Synthesizing Sound from Turbulent Field using Sound Textures for Interactive Fluid Simulation*. In *Proceedings of Eurographics*, pages 539–546. (Cited on page 13.)

S. Dubnov and N.Tishby [1997]. *Analysis of Sound Textures in Musical and Machine Sounds by means of Higher Order Statistical Features*. In *Proceedings of the International Conference on Acoustics Speech and Signal Processing*. Munich. (Cited on page 4.)

S. Dubnov, Z. Bar-Joseph, R. El-Yaniv, D. Lischinski, and M. Werman [2002]. *Synthesizing Sound Textures through Wavelet Tree Learning*. In *IEEE Computer Graphics and Applications*, pages 38–48. (Cited on pages 4, 6, 10, 13 and 14.)

J. Dunsby [1989]. *Considerations of Texture*. Music & Letters, 70(1), pages 46–57. (Cited on pages 2, 4 and 7.)

L. Engelen [2004]. *A rough guide to texture: oral physiology and texture perception of semi solids*. Proefschrift Universiteit Utrecht. (Cited on pages 2 and 3.)

J.-J. Filatriau and D. Arfib [2005]. *Instrumental Gestures and Sonic Textures*. In *Proceedings of the International Sound and Music Computing Conference SMC'05*. Salerno, Italy. (Cited on pages 4, 6, 7 and 13.)

J.-J. Filatriau, D. Arfib, and J.-M. Couturier [2006]. *Using Visual Textures for Sonic Textures Production and Control*. In *Proceedings of the International Conference on Digital Audio Effects (DAFx-06)*, pages 31–36. Montreal, Quebec, Canada. (Cited on pages 7 and 14.)

R. Fisher, Perkins, A. Walker, and E. Wolfart [2003]. *Hypermedia Image Processing Reference*. `http://homepages.inf.ed.ac.uk/rbf/HIPR2/kernel.htm`. Accessed on 27th Dec. 2006. (Cited on page 23.)

A. Flexer, E. Pampalk, and G. Widmer [2005]. *Novelty detection for spectral similarity of songs*. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR'05)*. (Cited on page 17.)

F. Fontana and R. Bresin [2003]. *Physics-Based Sound Synthesis and Control: Crushing, Walking and Running by Crumpling Sounds*. In *Proceedings on the XIV Colloquium on Musical Informatics (XIV CIM 2003)*. Firenze, Italy. (Cited on pages 12 and 13.)

J. Foote [1999]. *Visualizing Music and Audio using Self-Similarity*. In *Proceedings of ACM Multimedia '99*, pages 77–80. Orlando, Florida. (Cited on pages 18 and 21.)

J. Foote [2000]. *Automatic Audio Segmentation Using A Measure of Audio Novelty*. In *Proceedings of International Conference on Multimedia and Expo, ICME 2000*, pages 452–455. New York. (Cited on pages 24, 33 and 46.)

J. Foote and S. Uchihashi [2001]. *The Beat Spectrum: A New Approach To Rhythm Analysis*. In *Proceedings of International Conference on Multimedia and Expo, ICME 2001*, pages 881– 884. Tokyo, Japan. (Cited on pages 23 and 46.)

J. Gibson [1973]. *Die Sinne und der Prozesse der Wahrnehmung*. Verlag Hans Huber. (Cited on pages 4 and 6.)

H. Grassegger [2004]. *Phonetik Phonologie*. Schulz Kirchner Verlag. (Cited on page 26.)

C. M. Hamilton [2006]. *Machine Learning*. http://www.aaai.org/AITopics/html/machine.html. Accessed on 16th Oct. 2006. (Cited on page 23.)

P. Hanna, N. Lois, M. Desainte-Catherine, and J. Benois-Pineau [2004]. *Audio features for noisy sound segmentation*. In *Proceedings of International Conference on Music Information Retrieval (ISMIR)*. Barcelona. (Cited on page 6.)

R. Hoskinson [2002]. *Manipulation and Resynthesis of Environmental Sounds with Natural Wavelet Grains*. Master's thesis, The University of British Columbia. (Cited on pages 15 and 41.)

R. Hoskinson and D. Pai [2001]. *Manipulation and Resynthesis with Natural Grains*. In *Proceedings of the International Computer Music Conference ICMC'01*. Havana, Cuba. (Cited on pages 1, 5, 12, 15, 41, 42 and 44.)

T. Jehan [2005]. *Creating Music by Listening*. Phd thesis, Massachusetts Inst. Technology, Cambridge. (Cited on page 7.)

H. Lachenmann [1996]. *Klangtypen der neuen Musik*. Breitkopf und Härtel. (Cited on page 4.)

F. Liu [1997]. *Modeling Spatial and Temporal Texture*. Phd thesis, Massachusetts Inst. Technology, Cambridge. (Cited on page 4.)

B. Logan [2000]. *Mel Frequency Cepstral Coefficients for Music Modeling*. In *Proceedings of the First International Symposium on Music Information Retrieval (ISMIR)*. Plymouth, Massachusetts. (Cited on pages 18 and 19.)

L. Lu, L. Wenyin, and H. Zhang [2004]. *Audio Textures: Theory and Applications*. In *IEEE Transactions on Speech and Audio Processing*, pages 156–167. (Cited on pages 1, 7, 12, 15 and 33.)

A. Ma, F. Roters, and D. Raabe [2006]. *Simulation of textures and Lankford values for face centered cubic polycrystalline metals by using a modified Taylor model*. `http://www.mpie.de/1909/?tx_jppageteaser_pi1&type=1`. Accessed on 16th Oct. 2006. (Cited on page 2.)

M. Markou and S. Singh [2003]. *Novelty detection: A Review: Part 1: Statistical approaches*. Signal Processing, 12, pages 2481–2497. (Cited on page 23.)

N. E. Miner and T. P. Caudell [2005]. *Using Wavelets to Synthesize Stochastic Based Sound for Immersive Virtual environments*. In *ACM Transactions on Applied Perception*, pages 521–528. (Cited on pages 5 and 11.)

A. Misra, P. R. Cook, and G. Wang [2006]. *A New Paradigm for Sound Design*. In *Proceedings of the International Conference on Digital Audio Effects (DAFx-06)*, pages 319–324. Montreal, Quebec, Canada. (Cited on pages 5, 7, 10 and 13.)

J. G. Neuhoff [2004]. *Ecological Psychoacoustics*. Elsevier Academic Press, California, London. (Cited on page 4.)

F. Nielsen [2005]. *Visual Computing*. Charles River Media Inc, Massachusetts. (Cited on pages 9 and 10.)

M. J. Norris and S. Denham [2005]. *A sound texture detection algorithm*. In *JASA*, page 2613. (Cited on page 5.)

OGRE-team [2006]. *Materials-Ogre Wiki*. `http://grotsnik.ogre3d.org/wiki/index.php/Materials`. Accessed on 16th Oct. 2006. (Cited on page 3.)

D. O'Shaughnessy [2000]. *Speech Communications*. IEEE Press, New York. (Cited on pages 18, 20 and 33.)

E. Pampalk [2004]. *A Matlab Toolbox To Compute Music Similarity From Audio*. In *Proceedings of the First International Symposium on Music Information Retrieval (ISMIR 04)*. Barcelona. (Cited on page 31.)

G. De Poli, A. Piccialli, and C. Roads [1991]. *Representations of Musical Signals*. MIT Press, Massachusetts. (Cited on page 45.)

M. Puckette [2006]. *The Theory and Technique of Electronic Music*. (Cited on page 49.)

C. Roads [2004]. *Microsound*. The MIT Press. (Cited on page 11.)

D. Rocchesso and F. Fontana [2003]. *The Sounding Object*. Mondo Estremo. (Cited on page 12.)

N. Saint-Arnaud and K. Popat [1998]. *Computational Auditory Scene Analysis*. D. F. Rosenthal and H. G. Okuno, Lawrence Erlbaum Association, New Jersey. (Cited on pages 5 and 11.)

A. Schödl, R. Szeliski, D. H. Salesin, and I. Essa [2000]. *Video textures*. In *Siggraph 2000, Computer Graphics Proceedings*, page 3342. ACM SIGGRAPH, Addison Wesley Longman. (Cited on page 7.)

A. Schumacher and N. Balthasar [2006]. *Gotik Schrift Textur*. `http://demo.sfgb-b.ch/TG05/mittelalter/gotik/schrift/gotik_schrift_textur.htm`. Accessed on 16th Oct. 2006. (Cited on page 2.)

A. Di Scipio [1999]. *Synthesis of Environmental Sound Textures by Iterated Nonlinear Functions*. In *Proceedings of the 2nd COST g-6 Workshop on Digital Audio Effects DAFX'99*. Trondheim, Norway. (Cited on pages 4, 12 and 13.)

J. P. Sethna and P. A. Houle [1996]. *Acoustic Emission from crumpling paper*. In *Physics Review E*, pages 278–283. (Cited on page 13.)

J. M. Stowasser, M. Petschenig, and F. Skutsch [1994]. *Stowasser, lateinisch deutsches Schulwörterbuch*. öv et hpt, Vienna. (Cited on page 2.)

G. Strang and T. Nguyen [1997]. *Wavelets and Filterbanks*. Wellesley-Cambridge Press, Wellesley. (Cited on page 27.)

G. Strobl, G. Eckel, and D. Rocchesso [2006]. *Sound Texture Modeling: A survey*. In *Proceedings of Sound an Music Computing (SMC) International Conference*, pages 61–65. Marseille. (Cited on page 7.)

H. Tamura, S. Mori, and T. Yamawaki [1978]. *Textural features corresponding to visual perception*. In *IEEE Transactions on Systems, Man and Cybernetics*, pages 460–473. (Cited on page 2.)

A. S. Tanenbaum [1992]. *Modern Operating Systems*. Internals and Design Principles. Prentice-Hall, International. (Cited on page 10.)

R. Todd [2006]. *Compact Support*. `http://mathworld.wolfram.com/CompactSupport.html`. Accessed on 15th Dec. 2006. (Cited on page 42.)

J. Twicken [1998]. *Wavelet Basics*. `http://nova.stanford.edu/projects/sswrg/basics.html`. Accessed on 16th Oct. 2006. (Cited on page 41.)

G. Tzanetakis and P. Cook [2002]. *Musical Genre Classification of Audio Signals*. In *IEEE Transactions on Speech and Audio Processing,*, volume 10, pages 293–302. (Cited on page 33.)

G. Tzanetakis, G. Essl, and P. Cook [2001]. *Audio Analysis using the Discrete Wavelet Transform*. In *Proceedings of WSES International Conference Acoustics and Music: Theory and Applications (AMTA 2001)*. Skiathos, Greece. (Cited on pages 26 and 27.)

K. van den Doel [2005]. *Physically-based Models for Liquid Sounds*. In *ACM Transactions on Applied Perception*, pages 534–546. (Cited on page 13.)

Wikipedia [2006a]. *Discrete wavelet transform*. `http://en.wikipedia.org/wiki/Discrete_wavelet_transform`. Accessed on 16th Oct. 2006. (Cited on page 28.)

Wikipedia [2006b]. *Mel Scale*. `http://en.wikipedia.org/wiki/Mel_scale`. Accessed on 16th Oct. 2006. (Cited on pages 67 and 68.)

Wikipedia/de [2006]. *Utah-Teekanne*. `http://de.wikipedia.org/wiki/Utah-Teekanne`. Accessed on 16th Oct. 2006. (Cited on page 3.)

T. Wishart [1996]. *On Sonic Art*. Contemporary Music Studies. Harwood Academic. (Cited on page 5.)

X. Zhu and L. Wyse [2004]. *Sound Texture Modelling and Time-Frequency LPC*. In *Proceedings of the 7th International Conference on Digital Audio Effects DAFX'04*. Naples. (Cited on pages 5 and 11.)

U. Zölzer (Editor) [2002]. *Dafx: Digital Audio Effects*. John Wiley & Sons, Inc., New York, USA. (Cited on page 37.)