Audio Brush:
A Tool for Computer-Assisted Smart Audio Editing

C. G. v. d. Boogaart, R. Lienhart
Multimedia Computing Lab
University of Augsburg
86159 Augsburg, Germany
{boogaart,lienhart}@multimedia-computing.org

ABSTRACT

Starting with a novel audio analysis and editing paradigm, a set of new and adaptive audio analysis and editing algorithms in the spectrogram are developed and integrated into a smart visual audio editing tool in a "what you see is what you hear" style. At the core of our algorithms and methods is a very flexible audio spectrogram that goes beyond FFT and Wavelets and supports manipulating a signal at any chosen time-frequency resolution: the Gabor analysis and synthesis. It gives maximum accuracy of the representation, is fully invertible, and enables resolution zooming. Simple audio objects are localized in time and frequency. They can easily be identified visually and selected by simple geometric selection masks such as rectangles, combs and polygons. For many audio objects, however the structures in the spectrogram are rather complex. Therefore, we present several intelligent and adaptive mask selection approaches. They are based on audio fingerprinting and visual pattern matching algorithms. Spectrograms of individually recorded sounds under controlled conditions or interactively selected in the current spectrogram can be regarded as visual and sophisticated templates. We discuss how to generate templates, how to find the best match out of a database and how to adapt the match to the sound which we want to edit.

Categories and Subject Descriptors

H.5.5 [**Information interfaces and presentation**]: Sound and Music Computing—*Signal analysis, synthesis, and processing*; I.5.4 [**Pattern recognition**]: Applications—*Signal processing, Waveform analysis*

General Terms

Algorithms

Keywords

Visual audio editing, Audio fingerprinting, Gabor analysis, Audio objects

AMCMM'06, October 27, 2006, Santa Barbara, California, USA.

https://doi.org/10.1145/1178723.1178741

1. INTRODUCTION

Hearing, analyzing and evaluating sounds is possible for everyone. The reference-sensor for audio, the human ear, is of amazing capabilities and high quality. In contrast editing and synthesizing audio is an indirect and non-intuitive task needing great expertise. It is normally performed by experts using specialized tools for audio-effects such as a low-pass filter or a reverb. This situation is depicted in figure 1: A user can edit a given sound by sending it through an audio-effect (1). The input (2) and the output (3) are evaluated acoustically and sometimes but rarely also with a spectrogram (4,5). The audio-effects can only be controlled via some dedicated parameters (6) and therefore allow editing on a very abstract and crude level. In order to generate best results with this technique it is state of the art to record every sound separately on a different track under clean studio conditions. The effects can now be applied to each channel separately. More direct audio editing is desirable, but not yet possible.

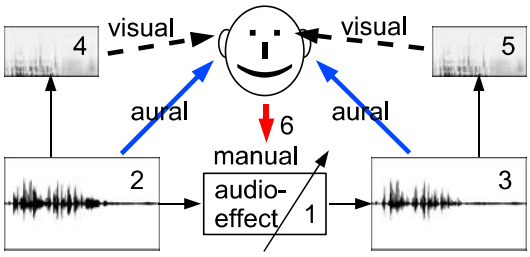

Figure 1: Classical situation in audio editing: A sound is sent through an audio-effect (1). The input (2) and the output (3) are evaluated acoustically and sometimes visually (4,5). The audio-effects are controlled via some dedicated parameters (6).

The goal of Audio Brush is to lower these limitations by providing a means to directly and visually edit audio spectrograms, out of which high quality audio can be reproduced. Figure 2 shows the new approach: A user can edit the spectrogram of a sound directly. The result can be evaluated either visually or acoustically, resulting in a shorter closed loop for editing and evaluating. This has several advantages:

1. A spectrogram is a very good representation of an audio-signal. Often speech-experts are able to read text out of speech-spectrograms. In our approach, the spectrogram is used as a representation of both, the

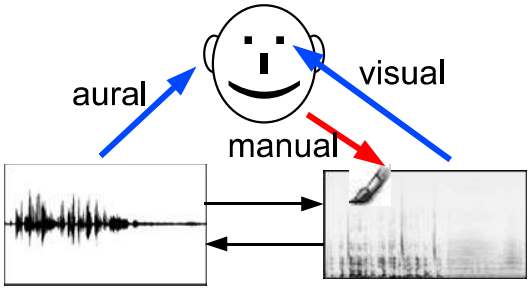

Figure 2: Editing with Audio Brush: The spectrogram of a sound is edited directly. The result can be evaluated either visually or acoustically.

original and the recreated audio-signal, which both can be represented visually and acoustically. It therefore narrows the gap between hearing and editing audio.

2. Audio is transient. It is emitted by a source through a dynamic process, travels through the air and is received by the human ear. It cannot be frozen for investigation at a given point in time and a given frequency band. This limitation is overcome by representing the audio signal as a spectrogram. The spectrogram can be studied in detail and edited appropriately before transforming it back into the transient audio domain.

Audio Brush allows to edit the spectrogram of a sound, which we call imaged sound, directly in the visual domain similar to editing bitmaps – a well understood paradigm as documented by standard software packages such as Adobe Photoshop™. Audio events are selected and modified with simple geometric masks. In order to assist the user in this process, we introduce a new paradigm in visual audio editing: Editing through the use of audio objects. Therefore sounds recorded beforehand under defined conditions are taken as audio objects and are stored in a database. They can be used to select and crop audio objects in the audio track. Once the audio objects are selected and separated, the audio track can be edited similar to a MIDI-track, something that is completely impossible in the time signal. We will report our approach and findings in this paper.

1.1 Related work

An approach of editing audio in the spectrogram has already been presented earlier. One implementation is Audiosculpt from IRCAM[1] followed by Ceres, Ceres2 and last by Ceres3[2], which are designed for musicians to create experimental sounds and also for education. Recently the software reNOVAtor[3] appeared. It is intended to clean a studio or live recording from short noise signals and needs an audio engineering expert as user. They all have in common, that they work as FFT/IFFT analysis/resynthesis packages, which allow editing the short time Fourier transformation spectrogram of an audio signal. The user has to choose several parameters for transformation and reconstruction, e.g. the window shape itself, but is restricted to very few fixed window lengths.

[1] www.ircam.fr
[2] music.columbia.edu/~stanko/About_Ceres3.html
[3] www.algorithmix.com/en/renovator.htm

Another approach is reported by Horn [13]. Is is based on auditory spectrograms, which model the spectral-transformation of the ear and is dedicated to speech, i.e. only for a small bandwidth. The transformation has no tunable parameters. The spectrogram is first abstracted to the so called part-tone-time-pattern and then edited and reconstructed.

There are two main differences of the earlier approaches compared to Audio Brush as we present it here: Firstly we use the Gabor transformation with a Gaussian window (see [11]). This transformation is optimal in terms of time-frequency resolution according to the Heisenberg uncertainty principle as well as in reconstruction quality. The uncertainty principle allows choosing the ratio of time to frequency-resolution. Therefore the user is allowed to choose this himself continuously and as only transformation parameter. In the following we refer to it as resolution zooming operation. Secondly, as imaged sounds are more or less complex structures, we introduce techniques for smart user assisted editing by the usage of template sounds. This helps to structure and handle imaged sounds, as they can be accessed as composition of audio objects very similarly to the events in a MIDI-track.

1.2 Contributions

The main contributions of this paper can be summarized as follows: We present a new selection paradigm for visual audio editing. Instead of simple geometric masks, audio objects out of a database are used. The best matching audio object is found automatically via a stable, flexible and fast detector, based on an enhanced audio fingerprinting system. We propose two technologies for applying the audio objects: first abstracting the template to a mask and secondly adapting the audio object through the use of an adaptive filter.

The paper is organized as follows: In Section 2 we summarize the design and general implementation issues of the Gabor transformation which we use for generating the spectrograms. In section 3 we summarize how imaged sounds can be edited manually through the use of simple geometric selection masks such as rectangles, combs and polygons. In section 4 we introduce the extension of visual audio editing based on audio objects. Section 5 concludes with a short summary of the main aspects.

2. FROM AUDIO TO VISUAL AUDIO AND BACK AGAIN

An audio signal is in general given in the 1D time-domain. In order to edit it visually, a 2D representation is necessary, which gives the user descriptive information about the signal and out of which the original or edited signal can be reconstructed. In this section we briefly discuss how to convert an audio signal into the image domain and back into the time domain. A spectrogram is used as image domain. It is calculated with a Gabor transformation (see [11], [9] and [10]) using a Gaussian window. The Gabor transformation allows for perfect localization in time and frequency according to the absolute bound expressed by the Heisenberg uncertainty principle. We use the term imaged sound to denote the spectrogram of an audio signal. It shows the magnitude of the audio signal. We choose one of multiple windows, to find the best matching time-frequency resolution for a given task. Further details regarding the Gabor

transformation and the choice of the best time-frequency resolution can be found in [3] and [2].

2.1 Fundamentals of the Gabor transformation

The Gabor transformation splits up a time function $x(t)$ in its time-frequency representation $X(t, f)$. As windowing function the Gaussian is used, which is given as: $g(t) = \frac{1}{\sqrt{2\pi\sigma_t^2}} e^{-\frac{1}{2}\frac{t^2}{\sigma_t^2}}$. Its Fourier transformation, i.e. its frequency function has the same Gaussian shape as the time function itself: $G(f) = \frac{1}{\sqrt{2\pi\sigma_f^2}} e^{-\frac{1}{2}\frac{f^2}{\sigma_f^2}}$ with $\sigma_f = \frac{1}{4\pi\sigma_t}$.

A Gabor system $g_{na,mb}(t)$ is derived by time shift a and frequency shift b:

$$g_{na,mb}(t) = e^{2\pi jmbt} g(t - na),\ n, m \in \mathbb{Z},\ a, b \in \mathbb{R}, \qquad (1)$$

a and b are called the lattice constants.

The Gabor transformation is defined as follows:

$$c_{nm} = X(na, mb) = \int_{-\infty}^{+\infty} x(t) g^*_{na,mb}(t)\, dt. \qquad (2)$$

The inverse transformation (reconstruction, Gabor expansion) is given as:

$$x(t) = \frac{1}{L} \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} c_{nm} g_{na,mb}(t). \qquad (3)$$

Details for the exact formulas: To ensure perfect reconstruction, the time frequency plane has to be oversampled. We therefore distinguish the critical sampled case $a_{crit} b_{crit} = 1$ from the oversampled case $a_{over} b_{over} = \frac{1}{5}$ and set $\frac{\sigma_t}{\sigma_f} = \frac{a_{crit}}{b_{crit}} = \frac{a_{over}}{b_{over}}$. The formulas (2) and (3) are for a continuous representation of $x(t)$ and calculate sums and integrals over infinity. For implementation they have to be discretized and the sums and integrals have to be truncated. The following definitions are needed: f_s sampling frequency. $M = \lceil \frac{1}{2} \frac{f_s}{b_{over}} \rceil$, $M \in \mathbb{N}$, t_{cut} half the window length, D decline of the Gaussian window from the maximum to the cut, $D = 30dB$. We get $t_{cut} = \sqrt{2ln\left(10^{\frac{D}{20}}\right)}\sigma_t$ and $N = t_{cut} f_s$, $N \in \mathbb{N}$. The Gabor transformation can then be implemented as:

$$c_{nm} = X(na, mb) = \frac{1}{L} \sum_{k=-N}^{N-1} x(kT) g^*_{na,mb}(kT) \qquad (4)$$

with its inverse:

$$x(kT) = \frac{1}{L} \sum_{n=\lfloor \frac{kT - t_{cut}}{a} \rfloor}^{\lceil \frac{kT + t_{cut}}{a} \rceil} \sum_{m=0}^{M-1} c_{nm} g_{na,mb}(kT) \qquad (5)$$

where $L = \sqrt{q \sum_{k=-N}^{N-1} |g(ka)|^2}$ and $m \in [0, M-1]$.

2.2 Resolution zooming

All time-frequency transformations are limited in their time-frequency resolution by the Heisenberg uncertainty principle, which says that the product of temporal resolution and frequency resolution has a total lower limit. The human ear itself has a time-frequency resolution, which closely reaches this limit and adapts its current time-frequency resolution to the current content of the signal in accordance to the Heisenberg uncertainty principle (see [1]). It is therefore advantageous also to adapt the resolution of the Gabor transformation to the current editing task. We refer to this operation as resolution zooming.

The time-frequency resolution is the only user relevant parameter regarding the generation of the spectrogram in Audio Brush. It is performed by choosing an adapted window length, which is equivalent to choosing the frequency shift b_{crit}, resulting in a time shift $a_{crit} = \frac{1}{b_{crit}}$ or vice versa.

Different choices reveal different characteristics of the signal. The properties of the 3D-space with the axes t, f and b_{crit} are clarified by the extremes of b_{crit}:

$b_{crit} \to \infty$**:** The window $g(t)$ becomes the Dirac impulse and the Gabor transformation becomes the time signal itself.

$b_{crit} = 0$**:** The window becomes $g(t) = const.$ losing its windowing properties and the Gabor transformation becomes the Fourier transformation.

Time-frequency resolution zooming allows increasing the resolution either in time or in frequency, while decreasing the resolution of the other domain.[4][5]

2.3 Visualization

The transformation-data is represented as complex numbers with real and imaginary 32 bit float values. These numbers cannot be visualized directly. Instead of complex numbers the magnitude values are used. They are then compressed and quantized to 8 bit unsigned integer values by calculating the square root of the values and scaling them (per image), to fit in 8 bit, while using the complete number range.

However, all processing tasks are performed on the transformation data, while the visualization is constantly updated. If a processing task is performed on the magnitude values only the phases are stored unchanged for the inverse transformation.

3. MANUAL AUDIO BRUSHING WITH GEOMETRIC SELECTION MASKS

We now describe how imaged sounds can be edited with geometric selection masks. It is performed similar to bitmap editing. Manual audio brushing serves as bases for the interactive editing with templates in the next section. More details on manual audio brushing can also be found in [3]. Figure 3 gives an overview over the several stages of editing. A time signal (1) is transformed (2) into one of a manifold of time-frequency representations (3). One representation

[4]Another conclusion of this discussion is that in contrast to images, the two axis time and frequency are not equivalent. A rotation of an image or of a region will lead to rather undesirable results and must be avoided.

[5]To achieve acceptable performance, the FFT (Fast Fourier Transform) can be used as an underlying technique to speed up the computation of the Gabor transformation. As the FFT exists only for some rare dedicated window lengths, this would restrict the possible time-frequency resolutions. In [2] we propose a solution of extending the FFT to arbitrary window lengths by introducing some computational overhead.

is chosen (4) and edited with help of a geometric selection mask (5). By inverse transformation (6) an edited time signal (7) is reproduced. By the appropriate choice of one of the manifold time-frequency representations, which refer to higher time or higher frequency resolution, it is possible to edit with high accuracy in time and frequency. If necessary, the process is repeated (8).

These techniques allow performing tasks, which are either very complicated or even impossible with classical filtering techniques. Bitmap editing operations serve as a basis for developing and understanding of content based audio manipulation techniques. Our tool also allows listening to selected regions of an imaged sound. This can be used to provide an instant feedback for the performed sound manipulations and thus serves as a perfect evaluation tool for the achieved sound quality.

3.1 Choosing a time-frequency resolution

The first task in editing is to zoom to the right time-frequency resolution. The right resolution allows to select a given sound very accurately. This will be illustrated with three prototypical sounds. Figure 4 shows the imaged sound of music with three instruments: guitar, keyboard and drums. The time-frequency resolution is set to $b_{crit} = 52.41Hz$. The sound of a cymbal is marked with a rectangle. Because of the chosen time-frequency resolution the cymbal-sound can be found very compactly in the spectrogram. The sound of the keyboard is represented very poorly (long gray horizontal parts at the bottom of the spectrogram). For the guitar, it is a good compromise (short dark black lines at the bottom, grouped in three groups, with many repetitions of the higher harmonics). For sounds with other characteristics different time-frequency resolutions have to be chosen. Figure 5 shows the imaged sound of clicks of a ball-pen with a time-frequency resolution set to $b_{crit} = 196.53Hz$, i.e. with a higher time resolution. This sound has mainly transient components and is strongly localized in time as can be seen clearly in the spectrogram. A third example illustrates that changing the time-frequency resolution not only changes the visualization, but also more clearly reveals or hides important information. See figure 6, which shows the sound of a piano playing a C-major scale, each note separately. The imaged sound is represented in three different time-frequency resolutions: $b_{crit} = 10.65Hz$, $b_{crit} = 49.13Hz$ and $b_{crit} = 196.53Hz$. For $b_{crit} = 10.65Hz$ it is easy to identify the fundamental frequency and the higher harmonics of each note. It is even possible to verify that a major scale and not a minor scale was played. By following the stairs of the first harmonic of each note, one can clearly see that the semitones are between III, IV and VII, VIII. For $b_{crit} = 49.13Hz$ the spectral structure of each tone can still be identified, but less accurate. The temporal decay of each harmonic can now be perceived separately. For $b_{crit} = 196.53Hz$ the temporal structure, how fast the notes were played, is emphasized, while the spectral structure is nearly completely smeared.

Zooming to a higher resolution on one axis reduces the resolution on the opposite axis. If the right time-frequency resolution is chosen, the sound qualities of interest are separated and can be selected separately. The original sound can be reconstructed from an imaged sound at any given time-frequency resolution. The combined time-frequency resolution is always at the total optimum. Time-frequency resolution zooming is therefore a strong feature of visual audio.

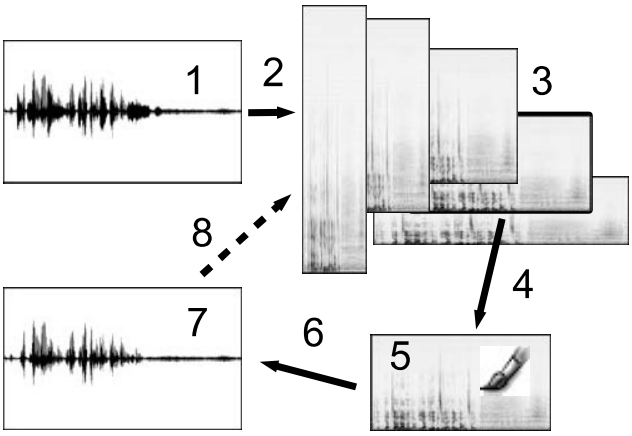

Figure 3: Overview of the several stages of editing: a time signal (1) is transformed (2) into one of a manifold of time-frequency representations (3). One representation is chosen (4) and edited with help of a geometric mask (5). By inverse transformation (6) an edited time signal (7) is recreated. If necessary, the process is repeated (8).

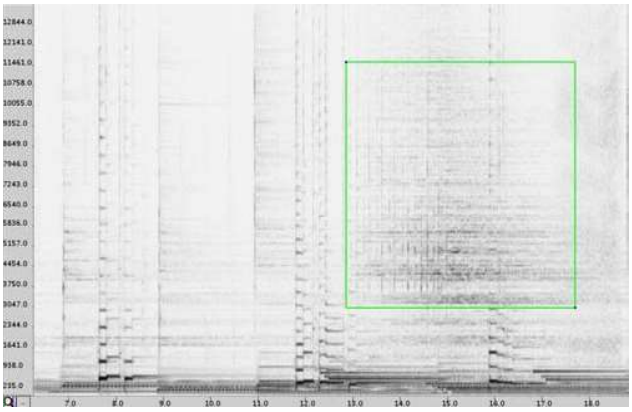

Figure 4: Music with three instruments: guitar, keyboard and drums. Time-frequency resolution: $b_{crit} = 52.41Hz$. The sound of a cymbal is marked with a rectangle.

Figure 5: Clicks of a ball-pen. Time-frequency resolution: $b_{crit} = 196.53Hz$. This sound has mainly transient components with very temporal characteristics.

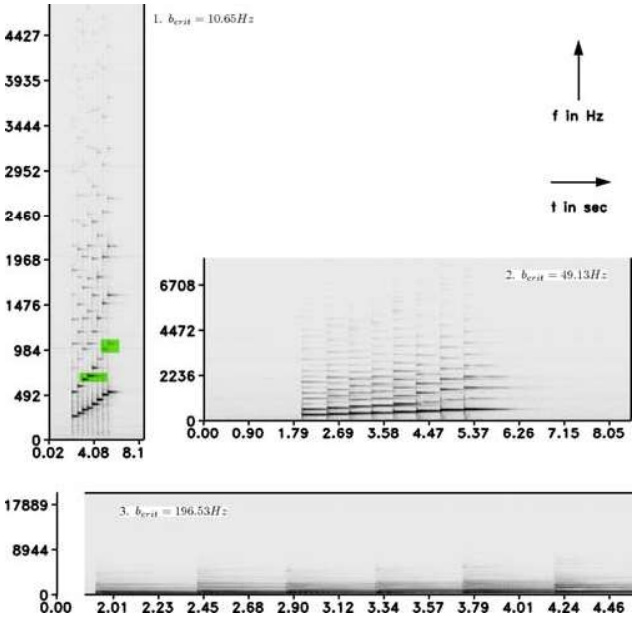

Figure 6: Sound of a piano playing a C-major scale, each note separately. Time-frequency resolutions: 1. $b_{crit} = 10.65Hz$, 2. $b_{crit} = 49.13Hz$ and 3. $b_{crit} = 196.53Hz$.

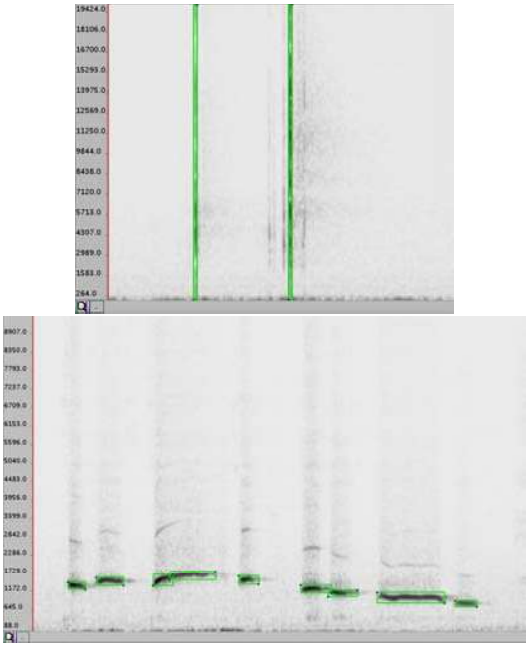

Figure 7: Rectangle mask for the two extreme shapes. Top ($b_{crit} = 196.53Hz$): Click of a ball-pen, rectangle mask with strong temporal characteristics. Bottom ($b_{crit} = 65.51Hz$): whistling, with strong tonal character.

3.2 Selection masks

Once you have selected a time-frequency resolution, the mask defining the sound pieces you want to edit must be constructed. In order to pick up different kinds of sounds, masks are necessary, which represent common structures of sounds. The better the mask matches a sound, the easier it is to select.

This already works with simple masks because of two reasons: Firstly, similar physical generation mechanisms of different sounds of the same class have roughly the same shape. Secondly, the ear is robust to small degenerations in sound quality due to experience (recall the bad sound quality of a telephone compared to original speech) and due to masking effects (see [15]) near the edge of a selecting mask.

We review some sensible selecting masks with corresponding sound examples:

- rectangle masks
- comb masks
- polygon masks
- combination masks

Rectangle mask: Figure 4 already contained an example of a rectangle mask. In this case the sound of a cymbal is such localized in the spectrogram ($b_{crit} = 52.41Hz$), that it can easily be isolated by a rectangle. This can be verified by listening to the outer or the inner part of the rectangle only.

The rectangle mask furthermore exists in two extreme shapes. One is useful in order to select sounds with very temporal characteristics, the other is useful for sounds with strong tonal character. Figure 7 shows an example of a click of a ball-pen (top) and an example of a whistling sound (bottom).

Comb masks: In contrast to the whistling sound, most tonal sounds not only incorporate the fundamental frequency, but also higher harmonics. This leads us to the comb masks. Figure 8 ($b_{crit} = 11.46Hz$) again shows the sound of a piano playing a C-major scale, each note separately. The in this case useful mask is called comb mask. As in this case a sound with a prominent pitch always generates a regular structure of higher harmonics which in its regularity is similar to a comb. A comb mask is defined by the following parameters: a function which describes the developing of the frequency of the highest harmonic, the number of harmonics and the bandwidth of every single harmonic. Figure 9 shows the spectrogram of a short piano piece ($b_{crit} = 11.46Hz$), which is of course polyphonic. The comb structure of each single note is nevertheless preserved. A single note can be selected and separated from the rest as it is illustrated.

Polygon masks: The last mask we want to explain simply uses polygons. Polygons allow to select such complex regions, while requiring more elaborateness of the user. They can be used e.g. to select structured and desired sounds between surrounding broadband noise.

Combination masks: Sounds can be very complex in the spectrogram. Thus it is possible to build up complex masks out of repeatedly applied simple masks of the same type or simple masks of different types.

3.3 Interaction

Once a sound of interest is selected with an appropriate mask, it is possible to edit the imaged sound. There are several useful possibilities.

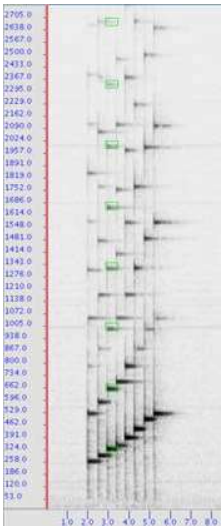

Figure 8: Sound of a piano playing a C-major scale, $b_{crit} = 11.46Hz$.

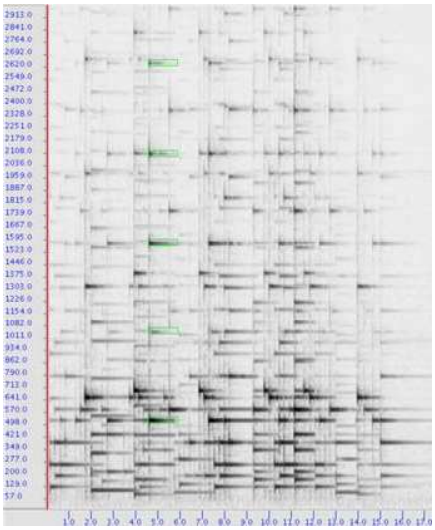

Figure 9: Sound of a piano playing a short polyphonic piece, $b_{crit} = 11.46Hz$.

Amplifying, stamping: The simplest editing is to multiply the magnitude values with a certain factor A. For $A = 0$ the sound is erased or stamped out, for $0 < A < 1$ the sound is damped in its level, for $A = 1$ it is unchanged and for $A > 1$ the sound is amplified.

Equalization: If a factor $A(f, t)$ as function of time and frequency is used, the result is similar to a very complex equalization process.

Cut, Copy&Paste: It is also possible to move the sound in the spectrogram. If only the time position is changed, the same sound is just moved or copied to another time instance.

Changing the pitch: If a sound is moved not only in time, but also in frequency, the pitch of the sound is changed, too.

Changing the shape: If the sound has harmonic components, in this case it is necessary to stretch or compress the sound in frequency direction, to preserve the harmonic structure.

Evaluating: A very helpful mechanism of Audio Brush is the playback of selected regions in the imaged sound. In practice it is not trivial to choose the correct shape for a mask for a given desired sound operation. The accuracy of the shape can however easily be evaluated by simply listening to the parts inside and the parts outside the mask respectively. By the presence or absence of a sound quality in these two parts of the sound, it can be clearly distinguished whether the shape of the mask has to be tuned further or whether it is already correct.

4. INTERACTIVE AUDIO BRUSHING WITH AUDIO OBJECTS

The examples above show that sounds from individual sound sources have distinctive shapes. Geometric masks can be used to describe such shapes. For instance the spectrogram of the piano notes in figure 9 shows that most of the sound energy is in the harmonics and can be selected with a comb mask. However, there are also nonharmonic parts, such as during the attack phase of every individual note. As these parts are spread over the whole frequency range and have no simple geometric structure, they cannot be handled with the techniques presented in section 3.

In this section we therefore introduce a new selection paradigm for visual audio editing based on audio objects. This approach is a much more flexible. Audio objects can adapt to very complex shapes, not to say to all kind of shapes, if they only can be recorded individually. In order to edit an audio track, matching audio objects are automatically selected out of a database. If a matching sound object from the database is found, it can be used as perfect matching mask.

Overview: First we discuss the features of audio objects in section 4.1. A sound recorded beforehand under defined conditions is used as an audio object. A database of audio objects serves as a toolbox for audio brushing.

Then matching audio objects have to be found in the database. This is described in section 4.2. Detection has to be performed resilient to typical variations in which audio objects can be experienced. The detection system has to be able to find not only exact matches, but also similar sounds, as not all possible sounds can be in the database.

Last but not least the audio object found has to be made fitting for the sound under consideration as described in section 4.3. The adapted audio object can then be applied to the sound track. See figure 10 for an overview of the procedure.

With this approach a track with a recording of an instrument can be handled nearly as flexible as if it were a MIDI-file. Editing an individual note in a MIDI-file is a two step procedure:

- Select the note.
- Adjust note number, velocity, beginning, length and all other parameters as desired.

In visual audio editing one more step is necessary:

- Remove the sound by subtracting an adapted audio object.

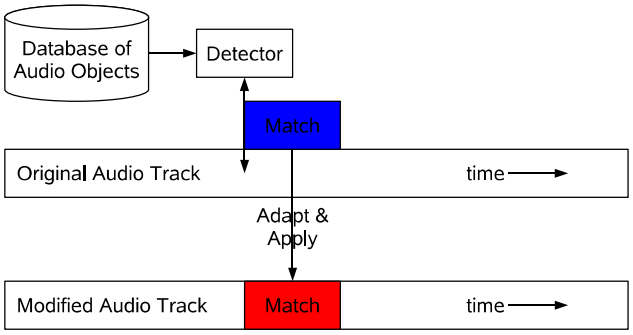

Figure 10: Overview of the interactive audio brushing with audio objects.

- Adjust pitch, volume, beginning, length and all other parameters of the adapted audio object as desired.
- Reinsert the audio object into the audio track.

Like copy&paste in MIDI-files, even notes can be added to an instrument track in visual audio editing:

- Find a note in the track, similar to the missing one.
- Adapt an audio object to the note.
- Adjust pitch, volume, beginning, length and all other parameters of the audio object as desired.
- Place the adapted audio object into its target location.

4.1 Audio Objects

Reproducible sounds, which are well-structured, can be treated as visual objects in the spectrogram. There they are characterized by a distinctive visual pattern – a pattern which looks similar even under typical variations such as pitch shifts, different play rates, and recordings from different microphones, different rooms, and playback devices. Examples could be individual notes played by an instrument, the sound of a closing car door, the rattling of a single key on a PC-keyboard or the click of a ball-pen, whose imaged sound we already discussed in figure 5 and figure 7. Another example is an individual note played by a piano. See for instance figure 11.

Not all possible sounds can be in the database, but typical sounds such as all common instruments. A General MIDI sound generator could serve as an automatic source generator for such a database.

Instead of audio objects recorded beforehand, the audio track itself can also serve as a template. If a recording was interfered by some noise, the interfered part can be marked and a similar but clean equivalent in the same recording can be found. The match can then be handled exactly the same way as a match from the database.

If the structure of a sound is stable over long times, even long portions of a sound can serve as audio-objects. A typical example is a single track of an audio CD.

Recordings of audio objects can be stored in a database of templates.

4.2 Audio object detection

This is the most crucial part of the system: the best matching audio object has to be found automatically. If

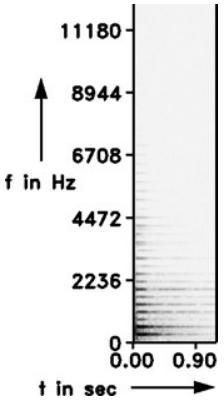

Figure 11: Example of a recording which can serve as audio object: A short C2 played as an individual note by a piano. $b_{crit} = 49.13Hz$.

this cannot be clearly decided, a short list of candidate audio objects has to be generated. As the number of templates increases, the flexible applicability of the system increases, too. Meanwhile, the system becomes too complex to be handled manually, because the user could not listen to all the templates in order to find the best match. With the audio object detection, this disadvantage becomes negligible. The system finds the best matching audio object itself or presents a short list of candidates. The main requirements for the automatic detection of audio objects are:

- Recognition resilient to typical variations and distortions.
- Finding similar audio objects, if exact matches are not available.
- Stable recognition even in the presence of interfering noise and other sounds.
- Fast detection even with a large database.

A brute force technique would be using the normalized cross-correlation between the audio object and all possible locations in the spectrogram. We derived our advanced algorithm from an audio fingerprinting system. This leads to a much more stable, flexible and faster algorithm. In this section we describe the resulting algorithm.

4.2.1 Algorithm overview

In audio fingerprinting one wants to solve the following problem: Given a database of clean recorded songs and a live audio stream. Is one of the database songs played in the live stream, and if yes, which one? As the conditions under which the live stream is produced are unknown, the audio fingerprinting algorithm has to be resilient to all kinds of degradation of the audio signal, such as distortions or noise.

For our purpose we have chosen the audio fingerprinting algorithm called "Distortion Discriminant Analysis" developed by Burges et al. (see [4] and [5]). This algorithm explicitly claims to recognize signals after several distortions. Most audio fingerprinting algorithms rely somehow on few heuristic or psychoacoustic features [6]. Distortion Discriminant Analysis relies only on spectral features, which are reduced in dimension, using the so called Oriented Principal

Component Analysis [7], maximizing the robustness of the information in the remaining coefficients.

The training data is provided in a clean version and the following distorted versions:

- Bass cut at 500Hz.
- Compander effect.
- Two bandstop filters: from 400Hz to 3400Hz and from 750Hz to 1800Hz.
- Filter simulating a telephone transmission with a bandpass from 200Hz to 3400Hz.
- Modified pitch, plus and minus 100 cent.
- Four different convolution effects with impulse responses, one for a large hall, one echo and two for small rooms.
- MP3 compression and decompression.

The convolutive distortions were calculated directly, the MP3 compression was done with lame[6]. All the other distortions were calculated with sox[7].

The input feature vectors during training and recognition are the complex coefficients of the Gabor transformation (the modulated lapped transformation was used in the original system). The Gabor transformation is calculated over the whole frequency range and then the values from 0Hz to 5kHz are passed on to the algorithm. Our algorithm is designed, to be invertible, i.e. the fingerprints can be reprojected into a time signal. Therefore the following modifications were applied to the original system: All preprocessing steps such as a psychoacoustic threshold are skipped. The modulus of the input data is not taken. All calculations are generalized from $\mathbb{R}$ to $\mathbb{C}$, thus the phases are preserved.

The system involves three parts: training of the dimensionality reduction, calculating the fingerprints of the audio object database and matching fingerprints of the database to fingerprints of the audio track.

In the following formulas vectors and matrices are printed in boldface. The asterisk $*$ denotes conjugate complex $\mathbf{x}^*$ of a vector $\mathbf{x}$ and $\mathbf{C}^*$ of a matrix $\mathbf{C}$. All matrices and vectors are complex. The only exception are the eigenvalues, which are real.

4.2.2 *Training of the dimensionality reduction*

The dimensionality reduction is mainly a modified PCA (principal component analysis). It explicitly trains robustness to distortions. As input we use different audio files, such as songs and audiobooks, as audio objects shall be recognized in all kinds of surrounding noise.

The whole system works in two stages, see figure 12. The input of the first stage are single feature vectors of one time instance. The input of the second stage are the combined output of several time instances of the first stage.

For training of each stage two runs over the complete training data are necessary, see figure 13.

In each stage in the first run the eigenvalues and eigenvectors are determined. The eigenvectors of the largest eigenvalues represent the principal components of the data. Therefore the dimensionality is reduced, by projecting the

[6] `lame.sourceforge.net`
[7] `sox.sourceforge.net`

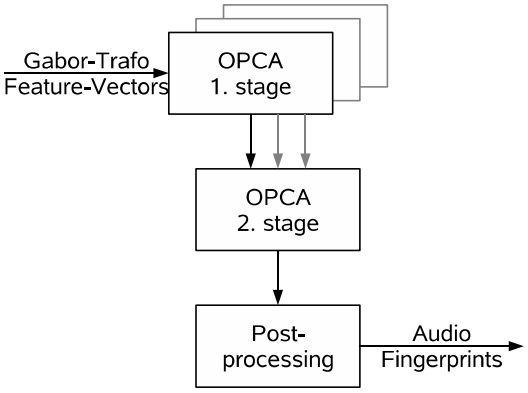

Figure 12: Scheme of the OPCA stages of the audio fingerprinter.

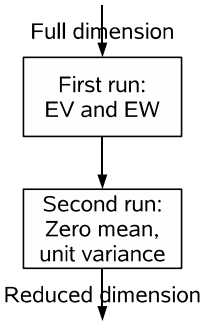

Figure 13: The two training runs for training of each OPCA stage.

input feature vectors of length n_{in} along the eigenvectors of the first n_{out} largest eigenvalues with $n_{out} < n_{in}$. The number of eigenvectors used to calculate the compressed vectors determine how much information is preserved. In the second run the projections of the input vectors along these eigenvectors are normalized. The aim is to have zero mean along the projections for the clean and distorted training data. Additionally the variance of the noise vectors (difference of clean and distorted training data) along that projection is normalized to one.

First trainings run of one stage: The first run over the training data is for both stages performed as follows: A covariance matrix $\mathbf{C}$ of the feature vectors $\mathbf{x}$ and a correlation matrix $\mathbf{R}$ of the noise vectors $\mathbf{z}^k$ is calculated. The noise vectors are defined as difference between distorted versions $\tilde{\mathbf{x}}^k$ of the feature vector and the clean feature vector $\mathbf{x}$: $\mathbf{z}^k = \tilde{\mathbf{x}}^k - \mathbf{x}$, where k marks one of N distortions. This gives:

$$\mathbf{C} = \frac{1}{m} \sum_i \left(\mathbf{x}_i - \left(\frac{1}{m} \sum_i \mathbf{x} \right) \right) \left(\mathbf{x}_i - \left(\frac{1}{m} \sum_i \mathbf{x} \right) \right)^* \quad (6)$$

and

$$\mathbf{R} = \frac{1}{m} \sum_i \frac{1}{N} \sum_k \mathbf{z}_i^k \left(\mathbf{z}_i^k \right)^*, \quad (7)$$

where N is the total number of distortions and m the total number of clean input vectors. The following generalized

eigenvalue problem has to be solved:

$$\mathbf{Cn} = \lambda \mathbf{Rn} \tag{8}$$

where $\mathbf{n}$ are the eigenvectors and λ are the eigenvalues. To solve the generalized eigenvalue problem, the matrix $\mathbf{R}$ has to be inverted. This is only possible if it has full rank, i.e. all eigenvalues of $\mathbf{R}$ are different. As this is not always the case, $\mathbf{R}$ has to be regularized before inverting. This can be done by adding a small fraction of the largest eigenvalue of $\mathbf{R}$ to its diagonal. The equation can then be solved with linear algebra packages such as lapack[8] together with blas[9]. The matrices have some special properties. As covariance respectively correlation matrices, they are positive semi definite and also hermitian. The generalized eigenvalue problem has therefore real eigenvalues and complex eigenvectors. The input vectors are then projected along the eigenvectors of the largest eigenvalues.

Second trainings run of one stage: In the second run the projections of the input vectors along these eigenvectors are normalized to have zero mean along the projections for the clean and distorted training data and unit variance of the noise vectors. Let $\mathbf{P}$ be the column-matrix of eigenvectors, i.e. $\mathbf{P} \in \mathbb{C}^{n_{in} \times n_{out}}$. The projection with reduced dimension of $\mathbf{x}$ along these eigenvectors is then calculated as the product $\mathbf{P}^T\mathbf{x}$. The normalization vectors are then calculated as:

$$\mathbf{o} = \frac{1}{m}\left(\sum_i \mathbf{P}^T\mathbf{x} + \frac{1}{N}\sum_i\sum_k \mathbf{P}^T\tilde{\mathbf{x}}^k\right) \tag{9}$$

and

$$\mathbf{s} = \sqrt{\frac{1}{mN}\sum_i\sum_k (\mathbf{P}^T\tilde{\mathbf{z}}^k)^2 - \left(\frac{1}{mN}\sum_i\sum_k \mathbf{P}^T\tilde{\mathbf{z}}^k\right)^2}. \tag{10}$$

The output of one stage: Finally the output $\mathbf{y}$ of a single stage is calculated as:

$$\mathbf{y} = (\mathbf{P}^T\mathbf{x} - \mathbf{o})/\mathbf{s}. \tag{11}$$

4.2.3 *Postprocessing*

The output of the second stage undergoes some postprocessing to generate finally the fingerprints. Each vector is normalized such that its L2-Norm is one. This compensates for different playback levels of the input audio data. The normalization factor is stored, in order to ensure continuity of the level of subsequent vectors. The distance to a previously random chosen set of 100 comparison vectors is calculated. A range factor is determined, which normalizes the mean of the L2-Norm of this differences to one. The range factor is also stored with the fingerprint and used during comparison. The fingerprints of the audio objects are placed in the database. The fingerprints of the audio track are computed on the fly.

4.2.4 *Matching the fingerprints*

The fingerprints of the database and of the audio track have to be matched. The euclidean distance, i.e. the L2-Norm of the difference is calculated and dived by the range factor of the database vector. If the distance is under a certain threshold, the audio object is a match.

[8] `www.netlib.org/lapack`
[9] `www.netlib.org/blas`

As this procedure requires calculating all differences, this is very time consuming. Goldstein et al. developed a fast indexing scheme called "Redundant Bit Vectors" to speed up the access, which also can be applied in our case (see [14]).

Determining the length: To reduce the number of mismatches, the threshold for the euclidean distance has to be chosen very low. As result an audio object matches only at some time instances and not over its whole length. In the surrounding of a match, the threshold can be increased, to explore the exact length of the match. Beginning from the initial match the subsequent fingerprint vectors of the audio track and the audio object are explored in positive and negative time direction. If the euclidean distance is under the increased threshold and so is the stored level normalization factor, then the audio object is continued. If not, the audio object is cut. Recall the template of a single piano note: if the initial match is at the attack of the note (every note has an attack), the remaining length of the note can be determined. If the audio object from the database is longer than the one from the audio track, it is just cut to the correct length. As the main energy of the sound is emitted during the attack and as the attack is critical for sound quality of an instrument, this produces negligible artefacts.

4.3 Apply

At the last step, the template can be applied to the signal, very similar to a mask from the preceding chapter. Possible editing tasks are: correcting the volume level, applying a selected equalization or deleting and afterward replacing the audio object.

As templates never match the audio object from the audio track exactly, they have to be adapted somehow. Several differences are possible, which have to be more or less compensated. At least the volume has to be adjusted, but also the recordings were normally made with a different reverberation or in a different room respectively. Perhaps only the type of instrument was the same (e.g. guitar), but not the instrument itself (a real wooden guitar and a MIDI-guitar-sound). That is, the template has to be made fitting for the sound.

For preprocessing the template, we present two alternatives. The first abstracts the template to a mask. The second estimates an adaptive filter, which corrects the following differences between the audio track (audio object + other sounds) and the template (clean audio object): phase, volume and spectral dissimilarities.

Abstracting to a mask: The first approach simply "stamps" out the audio object, i.e. the magnitude values are set to zero. This can be understood as generating a complex shaped mask out of a audio object. All magnitude values, which in the spectrogram of the audio object are larger than a certain frequency dependent threshold value, are declared to be inside the mask, the rest is outside the mask (see figure 14).

Estimating an adaptive filter: In contrast to visual objects, which are often nontransparent, audio objects are always additive, i.e. they shine through the energy of another audio object. The stamping approach, although attractive because of its simplicity and analogy to the visual domain, creates poorer results for increased overlapping of objects in the time-frequency domain.

Another method is subtracting a template from the audio track. As the template was recorded with a different

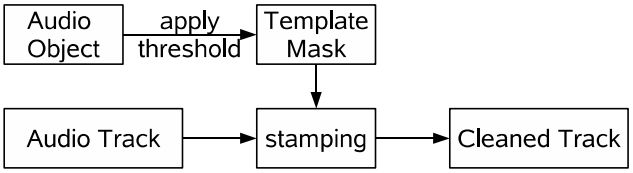

Figure 14: Scheme for stamping a detected audio object. Generating a template mask by applying a threshold and stamping the mask out of the spectrogram.

microphone and perhaps has a different level, it is first sent through a linear adaptive filter in order to match the audio object as well as possible before applying the difference. Figure 15 shows the scheme for this approach. An adaptive

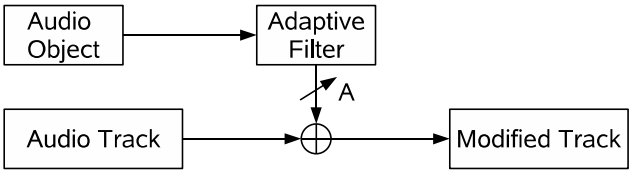

Figure 15: Scheme for erasing a detected audio object which is adapted beforehand with an adaptive filter. A is a factor. $A = -1$: the audio object is erased from the track. $-1 < A < 0$: the audio object is damped. $A = 0$: nothing changed. $A > 0$: the audio object is amplified.

filter is able to adapt phase, volume and spectral dissimilarities of the audio object to the signal. We use a simple FIR-filter with an LMS-update rule (see [8] and [12]). See figure 16 for the structure of the filter.

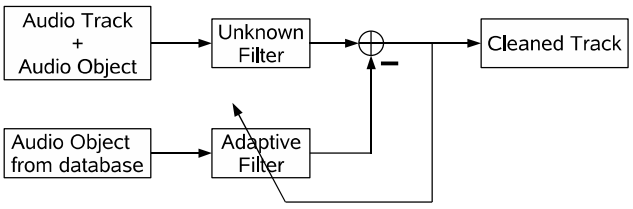

Figure 16: Structure of the adaptive filter. The adapted audio object is subtracted from the audio track.

5. CONCLUSION

A new tool for audio editing has been presented: Audio Brush. It allows editing audio in an intuitive and direct way. It opens up the possibility to freeze audio, which is naturally transient, and edit it in a static setting. This enables many new possibilities in terms of accuracy and flexibility not only in analyzing, but also in editing audio manually and automatically. We presented manual editing and extended it to smart user-assisted editing techniques by introducing the new paradigm of visual audio editing with audio objects. Audio objects greatly enhance the flexibility of achieving high quality editing results.

Possible further applications of the Audio Brush are: Improving the live recording of an instrument: All notes are deleted and replaced by a studio recorded version, e.g. to correct the tune of the instrument. Future developments could be to adopt image manipulation techniques, such as inpainting, for reconstructing erased or damaged audio parts. An interesting direction would be to develop techniques which would learn audio objects out of audio data, even if they are interfered by background noise.

Visual audio editing will unfold its full power by combining classical techniques with presented techniques in one tool.

6. REFERENCES

[1] C. G. v. d. Boogaart. *Master thesis: Eine signal-adaptive Spektral-Transformation für die Zeit-Frequenz-Analyse von Audiosignalen.* Technische Universität München, 2003.

[2] C. G. v. d. Boogaart and R. Lienhart. Fast gabor transformation for processing high quality audio. *ICASSP 2006, Toulouse, France*, 3:161–164, 2006.

[3] C. G. v. d. Boogaart and R. Lienhart. Visual audio: An interactive tool for analyzing and editing of audio in the spectrogram. In R. I. Hammoud, editor, *Interactive Video: Algorithms and Technologies.*, pages 107–130. Springer Verlag, 2006.

[4] C. Burges, J. Platt, and S. Jana. Distortion discriminant analysis for audio fingerprinting. *IEEE Transactions on Speech and Audio Processing*, 11(3):165–174, May 2003.

[5] C. J. C. Burges, J. C. Platt, and J. Goldstein. Identifying audio clips with rare. In *MULTIMEDIA '03: Proceedings of the eleventh ACM international conference on Multimedia*, pages 444–445, New York, NY, USA, 2003. ACM Press.

[6] P. Cano, E. Batlle, T. Kalker, and J. Haitsma. A review of algorithms for audio fingerprinting. *International Workshop on Multimedia Signal Processing, US Virgin Islands*, December 2002.

[7] K. Diamantaras and S. Kung. *Principal Component Neural Networks.* Wiley, 1996.

[8] P. S. R. Diniz. *Adaptive filtering: algorithms and practical implementation.* Kluwer Academic Publishers, Dordrecht, The Netherlands, 2002.

[9] H. G. Feichtinger and T. Strohmer. *Gabor Analysis and Algorithms: Theory and Applications.* Birkhäuser, Boston, 1998.

[10] H. G. Feichtinger and T. Strohmer. *Advances in Gabor Analysis.* Birkhäuser, Boston, 2003.

[11] D. Gabor. Theory of communication. *Journal of the Institution of Electrical Engineers*, pages 429–457, November 1946.

[12] S. Haykin. *Adaptive filter theory.* Prentice-Hall, Upper Saddle River, NJ, 2001.

[13] T. Horn. Image processing of speech with auditory magnitude spectrograms. *Acta Acustica united with Acustica*, 84(1):175–177, 1998.

[14] C. B. J. Goldstein, J.C. Platt. Redundant bit vectors for quickly searching high-dimensional regions. *Sheffield Machine Learning Workshop*, 2005.

[15] E. Zwicker and H. Fastl. *Psychoacoustics. Facts and Models.* Springer Verlag, second updated edition, 1999.