UNIVERSITÄT AUGSBURG
Universitätsbibliothek

OPEN ACCESS
OPUS AUGSBURG

Visual Audio: An Interactive Tool for Analyzing and Editing of Audio in the Spectrogram

C. Gregor van den Boogaart, Rainer Lienhart

Angaben zur Veröffentlichung / Publication details:

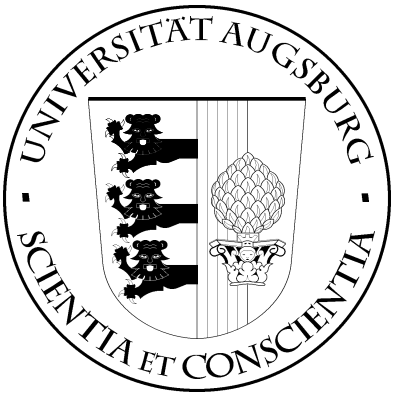
Boogaart, C. Gregor van den, and Rainer Lienhart. 2005. "Visual Audio: An Interactive Tool for Analyzing and Editing of Audio in the Spectrogram." Augsburg: Universität Augsburg.

Nutzungsbedingungen / Terms of use:

licgercopyright

Dieses Dokument wird unter folgenden Bedingungen zur Verfügung gestellt: / This document is made available under these conditions:
Deutsches Urheberrecht
Weitere Informationen finden Sie unter: / For more information see:
https://www.uni-augsburg.de/de/organisation/bibliothek/publizieren-zitieren-archivieren/publiz/

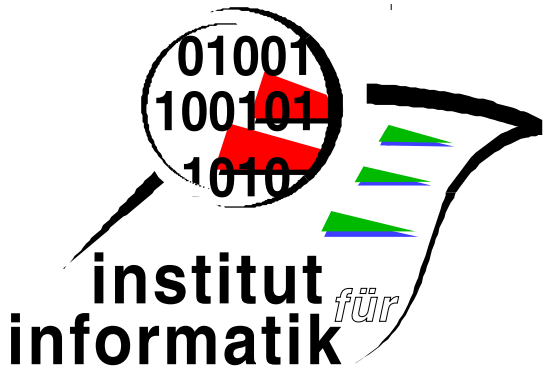
Universität Augsburg



Visual Audio: An Interactive Tool for Analyzing and Editing of Audio in the Spectrogram

C. G. v. d. Boogaart, R. Lienhart

Report 2005-22 December 2005



Institut für Informatik
D-86135 Augsburg

Copyright © C. G. v. d. Boogaart, R. Lienhart
Institut für Informatik
Universität Augsburg
D–86135 Augsburg, Germany
http://www.informatik.uni-augsburg.de
— all rights reserved —

Visual Audio: An Interactive Tool for Analyzing and Editing of Audio in the Spectrogram

C. G. van den Boogaart, R. Lienhart
Multimedia Computing Lab
University of Augsburg
86159 Augsburg, Germany
{boogaart,lienhart}@informatik.uni-augsburg.de

December 2005

We present a tool for analyzing and editing audio signals in the visual domain. As visual representation we use spectrograms, which give descriptive information about the sound. This allows analysing and editing audio in a "what you see is what you hear" style. Gabor analysis and synthesis serves as a basis to create images and recreate audio signals from the edited images in hi-fi quality.

As the structures in the spectrogram are rather complex, image processing and computer vision methods are applied for smart user assisted editing. Templates based on sounds recorded under defined conditions are therefore used. This allows detecting, separating, eliminating and/or modifying audio objects supervised (interactively) or automatically.

We further propose the usage of resolution zooming, to support manipulating the spectrogram of a signal at any chosen time-frequency resolution.

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, which needs 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 in clean studio conditions. The effects can now be applied to each channel separately. More direct audio editing is desirable, but not yet possible.

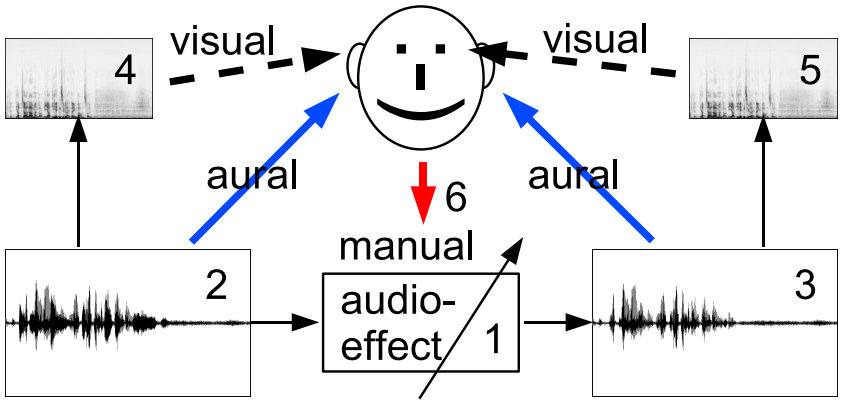


Figure 1: Classical situation in audio editing: A sound is sent through an audio-effect (1). The input (2) and the output (3) are evaluated acoustically and sometimes visually (4,5). The audio-effects are controlled via some dedicated parameters (6).

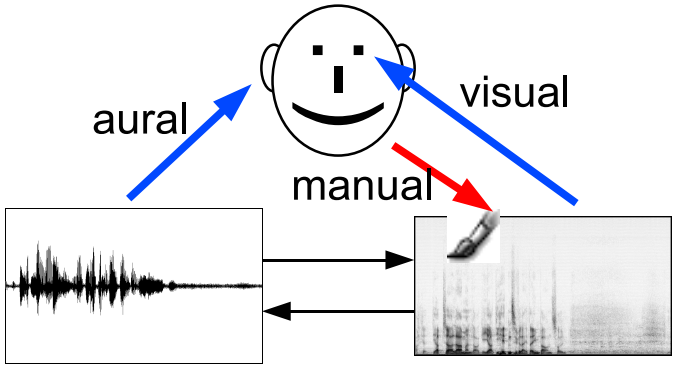


Figure 2: Visual Audio: The spectrogram of a sound is edited directly. The result can be evaluated either visually or acoustically.

The goal of Visual Audio 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 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 back into the transient audio domain.

Figure 3 gives an overview over the several stages of Visual Audio 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 (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). Figure 4 shows a screenshot of our

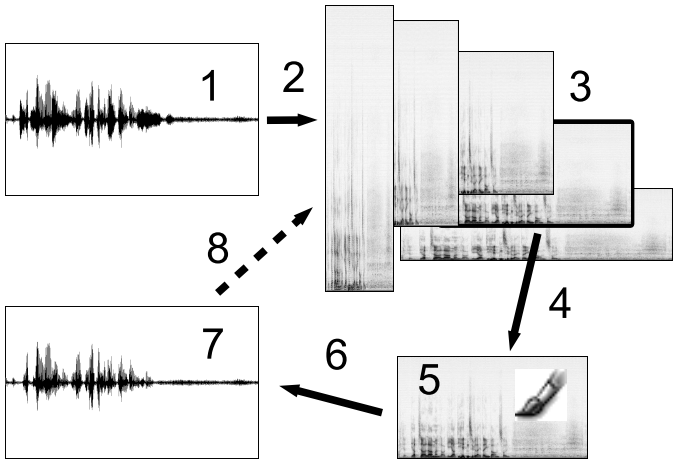


Figure 3: Overview of Visual Audio Editing: a time signal (1) is transformed (2) into one of a manifold of time-frequency representations (3). One representation is chosen (4) and edited (5). By inverse transformation (6) an edited time signal (7) is recreated. If necessary, the process is repeated (8).

implementation of Visual Audio Editing.

Related Work: Time-frequency transformations are a well-known and intensively used tool in automatic processing of audio signals. Standard transformations are: wavelets [4], [19], DFT and FFT [16] and Wigner-Ville-Distribution [17]. The Gabor transformation [8] is closely related to the MLT (Modulated lapped transform) [15], which has many applications in audio processing. The time-frequency transformations are mainly used either for calculating features or for compression and filtering. In the first case, the features are used for recognizing speech [13], speakers [18] or music pieces [9]. There is no way back from the features to the audio signal. An inverse transformation is not performed, but the features are used to derive higher

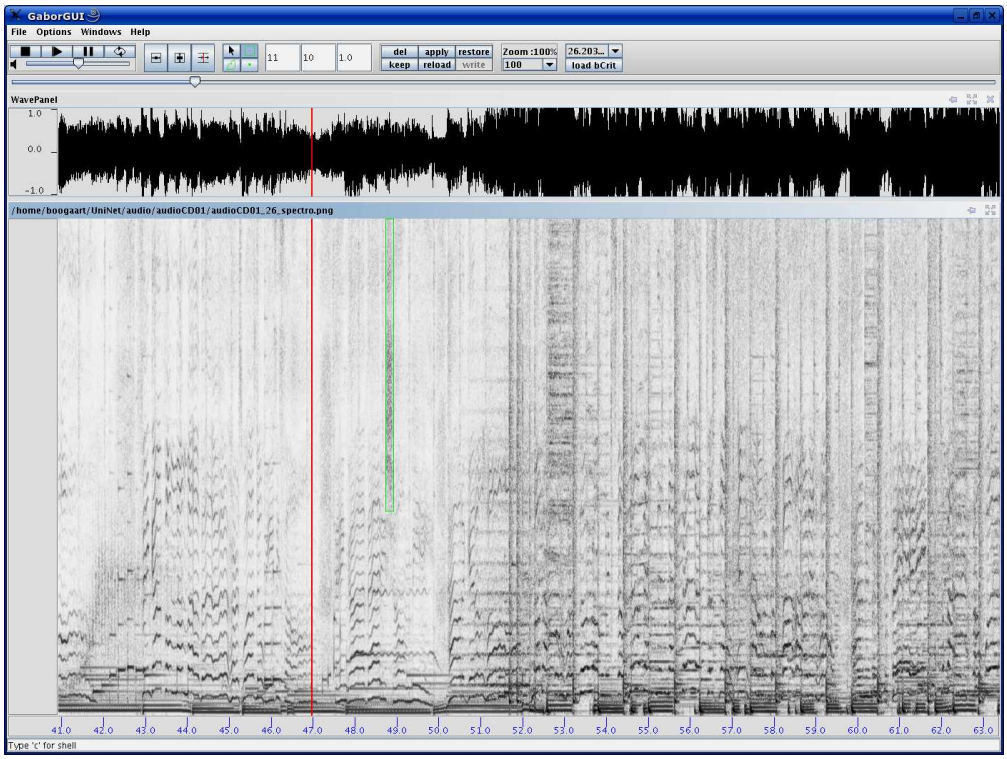


Figure 4: Screen shot of our implementation of Visual Audio Editing.

semantics from the audio signal. In the second case, e.g. lossy speech compression [19] or denoising [5], an inverse transformation is performed and the signal is either intentionally or unintentionally edited in the short term frequency domain. However the editing is not based on any visualization and thus no visual manipulation concepts can be applied.

Nevertheless the approach of editing audio in its spectrogram has already been presented earlier. One 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. 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. Another approach is reported by Horn [11]. 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 spectrogram is first abstracted to the so called part-tone-time-pattern and then edited and reconstructed.

There are two main differences compared to Visual Audio Editing as we present it here: First we use the Gabor transformation with the Gaussian window (see [8]). This transformation is optimal in terms of time-frequency resolution according to the Heisenberg uncertainty principle

[1] `http://www.ircam.fr`
[2] `http://music.columbia.edu/~stanko/About_Ceres3.html`

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. As sound images are more or less complex structures, we secondly introduce techniques for smart user assisted editing by the usage of template sounds, so called audio objects, which help to structure and handle the sound image.

This chapter is organized as follows: In Section 2 we explain how the Gabor transformation is used to create an image out of the audio signal and how to recreate the audio signal from the image. A single parameter, the resolution zooming factor, is used in order to adapt the time-frequency resolution to the task at hand. In Section 3 the manual editing of the image by the user is described. In Section 4 we discuss and give examples on smart user assisted editing. This is enabled by the use of audio objects, i.e. templates, which allow interacting with the image, preserving its complex structure and therefore preserving the high audio quality. 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 therefore discuss how to convert an audio signal into the image domain and how to recreate audio from that image domain.

2.1 Time-frequency transformation: Gabor transformation

As image domain we use a spectrogram. The spectrogram of an audio signal is called imaged sound. It shows the magnitude of a time-frequency transformation of the audio signal. Time-frequency transformations reveal local properties of a signal and allow to recreate the signal under certain conditions. The kind of revealed properties, however, depend strongly on the window and the window length. We use the Gabor transformation with a Gaussian window and apply multiwindow techniques to find the best matching window length for a given task (see e.g. [22]).

Fundamentals of the Gabor transformation: The Gabor transformation was introduced by Dennis Gabor [8] and has gained much attention in the near past (see e.g. [6] and [7]). In conjunction with the Gaussian window, which is not the only possible choice, the Gabor transformation has perfect time-frequency localization properties. It splits up a time function $x(t)$ in its time-frequency representation $X(t, f)$ and is defined as follows: From a single prototype or windowing function $g(t)$, which is localized in time and frequency, a Gabor system $g_{na,mb}(t)$ is derived by time shift a and frequency shift b (see [20]):

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

a and b are called the lattice constants. The Gabor system covers the whole time-frequency

plane. The Gabor transformation is then expressed as follows:

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

c_{nm} are called the Gabor coefficients of $x(t)$. The inverse transformation or Gabor expansion is calculated with the window function $\gamma(t)$, which is called dual window of $g(t)$. With the Gabor system $\gamma_{na,mb}(t)$ defined analogously to equation (1), the Gabor expansion is defined as follows (see [1]):

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

with $L = \sum_{k=-\infty}^{\infty} |\gamma(ka)|^2$. The theory of the Gabor transformation leads to the result, that the window functions $g(t)$ and $\gamma(t)$ and the lattice constants a and b must fulfill certain requirements in order to assure the invertibility (see [7]). The appropriate choices are discussed in the following two paragraphs. For use in digital signal processing formulas (2) and (3) have to be discretized, using sums instead of integrals and sums of finite length, which is discussed in the last paragraph of this section.

The Gaussian window: We start with the discussion of the Gaussian window function, which is given as:

$$g(t) = \frac{1}{\sqrt{2\pi\sigma_t^2}} e^{-\frac{1}{2}\frac{t^2}{\sigma_t^2}} \tag{4}$$

and has the following advantageous properties (see e.g. [23]):

- Minimal extent in the time-frequency plane according to the Heisenberg uncertainty principle.
- Localized shape, i.e. only one local and global maximum with strict decay in time and frequency direction.

The uncertainty principle of Heisenberg says that the product of temporal and frequency extent of a window function has a total lower limit. If the extents are defined in terms of standard deviations of the window function and of its Fourier transformation respectively, it can be expressed with the following inequality (see [19]):

$$\sigma_t \sigma_f \geq \frac{1}{4\pi}\,. \tag{5}$$

The "$=$" is only reached for the Gaussian window function (see e.g. [21]), which therefore has the minimal possible extent in the time-frequency plane. Its Fourier transformation has the same Gaussian shape as the time function itself:

$$G(f) = \frac{1}{\sqrt{2\pi\sigma_f^2}} e^{-\frac{1}{2}\frac{f^2}{\sigma_f^2}}, \text{ with } \sigma_f = \frac{1}{4\pi\sigma_t}. \tag{6}$$

The dual window $\gamma(t)$ should also be localized in time and frequency to preserve the local influence of the Gabor coefficients on the result of the inverse transformation. Also this is best achieved by the Gaussian as dual window-function. To ensure perfect reconstruction some restrictions are imposed on the choice of lattice constants a and b as discussed in the following paragraph.

Choice of lattice constants and oversampling factor: In his original paper [8] Dennis Gabor suggested to choose $ab = 1$ which is called critical sampling. This choice has implicit influence on the shape of the dual window. In fact with the Balian-Low theorem it can be shown that in this case the dual window extends to infinity and is not localized at all (see [7]). The solution is to choose $ab < 1$, which is referred to as the oversampled case. It leads to better localized dual windows and numerically stable analysis and synthesis.

We therefore have to determine an appropriate oversampling factor for $ab < 1$. In the literature normally the cases of rational oversampling ($ab = \frac{p}{q}$, $p, q \in \mathbb{N}$ and $p < q$) and integer oversampling ($ab = \frac{1}{q}$, $q \in \mathbb{N}$) are discussed. Bastiaans [1] proposes taking $ab = \frac{1}{3}$ for which the ideal dual window of the Gaussian is very similar to a Gaussian window. He mentions that for increasing values of q the resemblance of the Gaussian window and its dual window further increases. In simple empirical hearing tests we found that an oversampling factor starting at $ab = \frac{1}{5}$ avoids hearable differences between an original and a reconstructed sound in case of using the same Gaussian window for analysis and synthesis. This holds for speech and for full bandwidth music. An oversampling factor of $q = 5$ is therefore the necessary and sufficient accuracy for high quality audio processing.

In the following, if necessary we name a, b for the critical sampled case a_{crit}, b_{crit} and for the oversampled case a_{over}, b_{over}. The resulting lattice and how it covers the time-frequency plain is illustrated in figure 5. The gray shaded circles indicate the extent of a single Gaussian window in the time-frequency plain expressed in its standard deviations σ_t and σ_f. To ensure the same

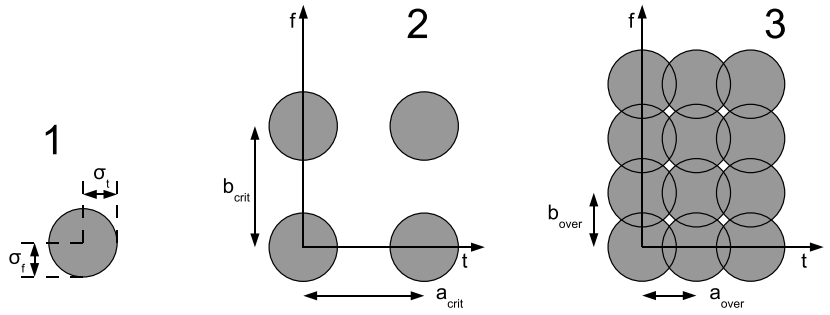


Figure 5: Coverage of the time-frequency plain: The gray shaded circles indicate the extent of a (1) single Gaussian window expressed in its standard deviations σ_t and σ_f, (2) the critically sampled case and (3) the oversampled case.

overlapping of the Gaussians in time and frequency direction, we therefore have to set:

$$\frac{\sigma_t}{\sigma_f} = \frac{a_{crit}}{b_{crit}} = \frac{a_{over}}{b_{over}}. \tag{7}$$

With $a_{over}b_{over} = \frac{1}{q}$ this holds for:

$$a_{over} = \frac{a_{crit}}{\sqrt{q}}, b_{over} = \frac{b_{crit}}{\sqrt{q}}. \tag{8}$$

With formula (6) and formula (7) we can solve:

$$\sigma_t = \frac{1}{\sqrt{4\pi}}\sqrt{\frac{a}{b}}, \sigma_f = \frac{1}{\sqrt{4\pi}}\sqrt{\frac{b}{a}}. \tag{9}$$

As in the following we will always choose b and determine the other values, we continue with $ab = \frac{1}{q}$:

$$\sigma_t = \frac{1}{\sqrt{4\pi q}}\frac{1}{b}, \sigma_f = \sqrt{\frac{q}{4\pi}}b. \tag{10}$$

With an oversampling factor of $q = 5$ and these formulas, it is feasible to take the Gaussian window $g(t)$ as its own dual window $\gamma(t) = g(t)$. One still has the freedom to choose either a_{crit} or b_{crit}, i.e. to choose an appropriate window length for the current task.

Discretization and Truncation: The formulas (2) and (3) are for a continuous representation of $x(t)$ and calculate sums and integrals over infinity. Therefore they have to be discretized and the sums and integrals have to be truncated in order to be implemented. This corresponds to bandlimiting and sampling $x(t)$ and to truncating $g(t)$. $x(t)$ is sampled with the sampling frequency f_s. With $T = 1/f_s$ and $k \in \mathbb{Z}$ $x(t)$ becomes $x(kT)$. To fulfill the sampling theorem, the bandwidth f_B of $x(kT)$ has to fulfill $f_B \leq f_s/2$ (see e.g. [16])[3]. We define $M \in \mathbb{N}$ as the number of frequency bands with $M = \lceil \frac{1}{2}\frac{f_s}{b_{over}} \rceil$. With t_{cut} half the window length, we define $N \in \mathbb{N}$ with $N = t_{cut}f_s = \frac{t_{cut}}{T}$ as half the window length in samples. The Gabor transformation can then be implemented as:

$$c_{nm} = X(na, mb) = \frac{1}{L}\sum_{k=-N}^{N-1} x(kT)g^*_{na,mb}(kT) \tag{11}$$

with its inverse:

$$x(kT) = \frac{1}{L}\sum_{n=\lfloor \frac{kT-t_{cut}}{a} \rfloor}^{\lceil \frac{kT+t_{cut}}{a} \rceil}\sum_{m=0}^{M-1} c_{nm}g_{na,mb}(kT) \tag{12}$$

where $L = \sqrt{q\sum_{k=-N}^{N-1}|g(ka)|^2}$ and $m \in [0, M-1]$. We still have to determine t_{cut} and N respectively, which correspond to the truncation of the Gaussian window as already mentioned. We have chosen empirical listening tests to find an appropriate truncation. The goal was to get a reproduced sound with no hearable difference from the original sound. We define the decline D

[3]For hi-fi audio typical values are $f_s = 44.1kHz$ (CD) or higher and $f_B = 20kHz$.

of the Gaussian window from the maximum to the cut expressed in dB as:

$$D = 20log\frac{g(0)}{g(t_{cut})}dB. \tag{13}$$

For a given D we get with (4):

$$t_{cut} = \sqrt{2ln\left(10^{\frac{D}{20}}\right)}\sigma_t. \tag{14}$$

Values of $D \geq 30dB$ have shown to be completely sufficient for high quality audio.

Remark: Storing the Gabor coefficients is very efficient independent of the time-frequency resolution. A discretized time signal of duration t_{dur} needs $N^{\mathbb{R}} = t_{dur}f_s$ real sample values. The Gabor coefficients need

$$N^{\mathbb{C}}_{Gabor} = \frac{t_{dur}}{a}\frac{f_B}{b} = t_{dur}f_Bq \tag{15}$$

complex or

$$N^{\mathbb{R}}_{Gabor} = 2 \cdot N^{\mathbb{C}}_{Gabor} = 2t_{dur}f_Bq \tag{16}$$

real and imaginary values combined for storage. In summary $N^{\mathbb{R}}_{Gabor} = q \cdot N^{\mathbb{R}}$ values independent of the current time-frequency resolution are needed to store the Gabor transformation. This can be further enhanced by setting $f_B = 20kHz$ with $f_B < f_s/2$.

2.2 One degree of freedom: resolution zooming

The human ear has a time-frequency resolution, which closely reaches the physical limit expressed in the Heisenberg uncertainty principle . It furthermore adapts its current time-frequency resolution to the current content of the signal according to the Heisenberg uncertainty principle (see [2]). It is therefore advantageous also to adapt the resolution of the transformation to the current editing task. The resolution zooming feature is discussed in this section.

Heisenberg uncertainty principle: We already applied the Heisenberg uncertainty principle to the functions of the window and the dual window (see section 2.1). It also holds for the function of the signal, which of course in general has a worse resolution than the theoretical limit.

As the Gabor transformation is a discretized version of the STFT, we can discuss the continuous case of the STFT. The STFT can be expressed as a convolution of signal $x(t)$ with $h^*(-t, f) = w^*(-t)e^{-j2\pi ft}$:

$$X(t, f) = x(t) * h^*(-t, f) = \int_{-\infty}^{+\infty} x(\tau)h^*(-(t-\tau), f)\, d\tau. \tag{17}$$

This is similar to adding the variances of two statistically independent random variables X and Y, which form a new random variable $Z = X + Y$. Their probability density functions are also convolved and the variance of Z is then given by (see [10]):

$$\sigma^2_Z = \sigma^2_X + \sigma^2_Y. \tag{18}$$

Therefore the time and frequency variances of a signal and a window are added in the form:

$$\sigma^2_{t_{transformation}} = \sigma^2_{t_{signal}} + \sigma^2_{t_{window}}, \tag{19}$$

$$\sigma^2_{f_{transformation}} = \sigma^2_{f_{signal}} + \sigma^2_{f_{window}}. \tag{20}$$

Consequently the achievable accuracy of editing a given signal is given by the superposition of the signal's and window's uncertainty. As time and frequency resolution are interconnected, one has to give up time resolution in order to improve the frequency resolution and vice versa.

Which window length to choose: So one has to choose the frequency shift b_{crit} resulting in a time shift $a_{crit} = \frac{1}{b_{crit}}$ or vice versa. This is equivalent to choosing an adapted window length. Different choices lead to different time-frequency representations of the same signal in the 3D-space with the axes t, f and e.g. b_{crit}. Although the same signal is represented, different characteristics of the signal are revealed in different layers with constant b_{crit}. The properties of this space can be 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.

In Visual Audio Editing it is therefore essential to calculate the Gabor transformation of an audio signal with different choices of b_{crit} and to perform the respective editing task in the layer $b_{crit} = const.$ which allows the best accuracy for the current task. This can be understood as zooming, which allows increasing the resolution of the representation of a signal either in time or in frequency, while the resolution of the other domain decreases.

From this discussion it also follows, 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 has to be omitted.

Example: Figure 6 a) shows a typical spectrogram of a speech signal. The spectrogram is calculated with $b_{crit} = 64Hz$, which results with $b_{over} = 28.6Hz$ in roughly 699 frequency bands in the range from $0Hz$ to $20kHz$. One property of the signal is hidden in this spectrogram: The recording was accidentally interfered by power line hum at $50Hz$ (common in Europe). The hum can be heard, if the sound is played at higher volume levels, but it cannot be seen in this spectrogram. Figure 6 b) shows a spectrogram of the same recording with a much higher frequency resolution of $b_{crit} = 5Hz$, i.e. $b_{over} = 2.24Hz$. In this spectrogram it is easy to distinguish the hum at $50Hz$ and his higher harmonics from the rest of the signal. Figure 6 c) shows the spectrogram with $b_{crit} = 64Hz$ again, but the image is "stretched" in frequency direction. The energy of the hum is distributed widely in the spectrogram and the higher harmonics are totally blurred and cannot be recognized at this frequency-resolution.

2.3 Fast computation

The Gabor transformation as we use it here can be expressed as a special form of the DFT (Discretized Fourier Transform), if the window is understood as a part of the signal. At every

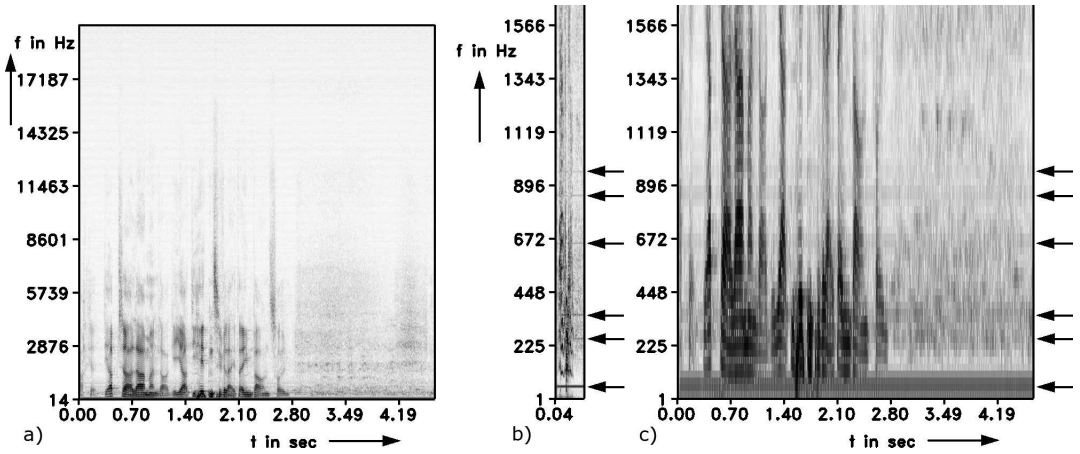


Figure 6: Typical spectrograms for speech signal with an interfering power line hum. a): $b_{crit} = 64Hz$, b): $b_{crit} = 5Hz$, c): $b_{crit} = 64Hz$, "stretched" in frequency direction. For convenience b) and c) are cut above $1600Hz$. The interfering power line hum and higher harmonics can be recognized easily in the middle spectrogram. They are indicated by arrows on the right at $50Hz$, $250Hz$, $350Hz$, $650Hz$, $850Hz$ and $950Hz$. The energy of the hum is distributed widely in the right spectrogram and the higher harmonics are totally blurred.

multiple of the time shift a_{over} a DFT has to be computed. For hi-fi audio with high sampling frequency and some signal length, this leads to rather bad performance expressed in computation time. The DFT can be faster implemented as FFT (Fast Fourier Transform), if the the DFT length is a power of 2. As the DFT as then restricted to dedicated lengths, these forces the use of only some $b_{over} = \frac{f_s}{2M}$. They are moreover dependent on the sampling frequency f_s of the original signal. As for Visual Audio Editing b_{over} shall be chosen continuously the only possible solution up to now was to calculate a much more time consuming DFT.

We propose a solution to soften the hard restriction by the FFT window length, which allows to take advantage of the higher performance of the FFT. This introduces some computational overhead, which is generally acceptable if the overall execution time is still lower than that of a DFT, which is often the case.

For fast calculation of a DFT it is common practice to zero pad a given signal to a power of two boundary. By this action, the frequency spacing of the resulting values is also modified. We follow a very similar idea. By extending the window from a FFT boundary at $2M$ to a broader window at the FFT boundary $2\dot{M} = 2M \cdot 2^{\eta}$ ($\eta \in \mathbb{N}$), the density of frequency values is increased. Instead of $b_{over} = \frac{f_s}{2M}$ we get $\dot{b}_{over} = \frac{f_s}{2\dot{M}} = \frac{b_{over}}{2^{\eta}}$. Because of the frequency-shape of the window, which is not altered by altering the FFT length, the frequency values can now be downsampled by a factor $\kappa \in \mathbb{N}$, $\kappa < 2^{\eta}$, in order to get a $\ddot{b}_{over} = \kappa \dot{b}_{over}$. Figure 7 explains the scenario.

Before performing the inverse transformation the skipped values have to be reconstructed. This can be done in the (complex) frequency signal very similar to the way it is done with (real)

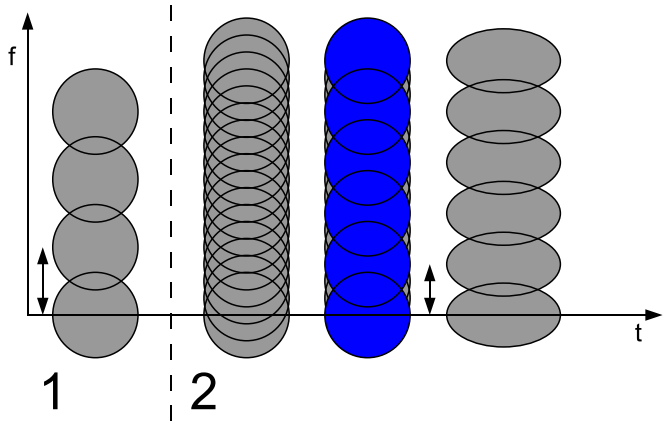


Figure 7: This illustrates the downsampling. The vertical arrows mark in each case b_{over}. (1) shows a FFT with length $2M$ and the corresponding b_{over}. (2) shows a FFT with length $2\dot{M} = 2M \cdot 2^2$. The blue circles mark a frequency vector generated by downsampling with factor 3. This alters the lattice constants. The resulting frequency vector with adapted σ_t and σ_f is shown.

time signals: sample up by the factor κ and filter with the reconstruction filter. This filter is in our case the frequency representation of the Gaussian window (see formula (6)). As a convolution in the frequency domain is a multiplication in the time domain, this step can be done much more efficient after the inverse transformation in the time domain by multiplication with the time representation of the Gaussian window. Performance measurements and more details can be found in [3].

2.4 Visualization: magnitude, phase and quantization

The transformation-data is represented as complex numbers with real and imaginary 32 bit float values. These numbers can not 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.

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

3 Manual audio brushing

We now describe, how imaged sounds can be edited manually. An obvious approach is to edit them like bitmaps – a well understood paradigm as documented by standard software packages such as Adobe Photoshop^{TM}. These techniques will allow us to perform 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 to listen 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 Time-frequency resolution

The first task in editing is to find 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 8 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 compact in the spectrogram. For sounds with other characteristics different time-frequency

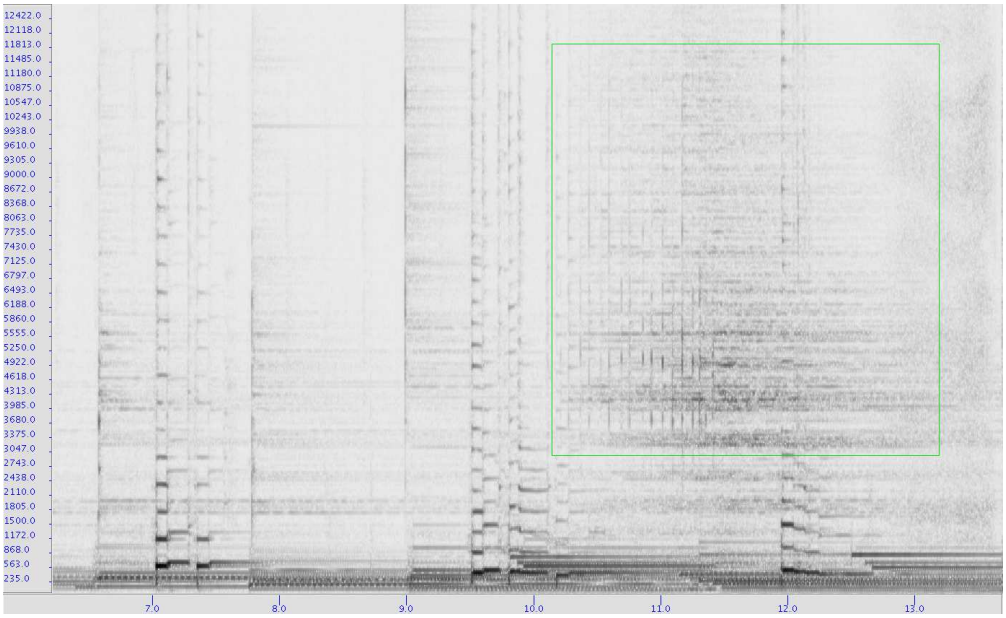


Figure 8: Music with three instruments: guitar, keyboard and drums. Time-frequency resolution: $b_{crit} = 52.41Hz$. The sound of a cymbal is marked with a rectangle.

resolutions have to be chosen. Figure 9 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 very 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 10, 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: $b_{crit} = 10.65Hz$, $b_{crit} = 49.13Hz$

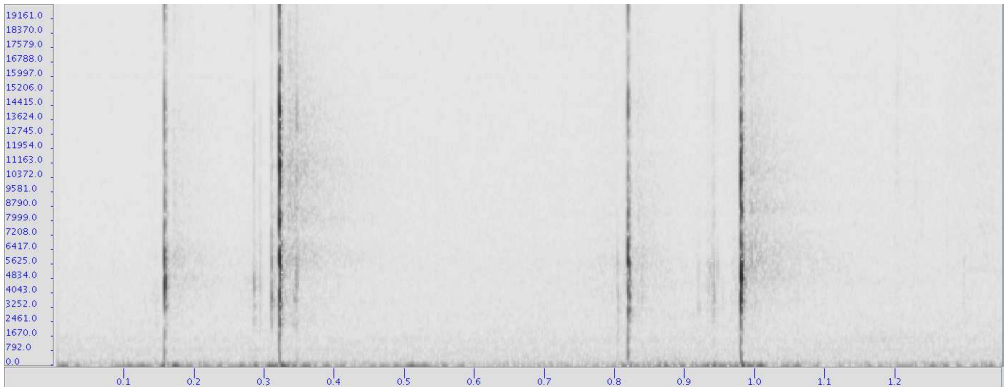


Figure 9: Clicks of a ball-pen. Time-frequency resolution: $b_{crit} = 196.53Hz$. This sound has mainly transient components with very temporal characteristics.

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 is still to identify, 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 still 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.

3.2 Selection masks

Once you have selected the time-frequency resolution, a 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 [24]) near the edge of a selecting mask.

We review some sensible selecting masks with corresponding sound examples.

- rectangle masks
- comb masks
- polygon masks

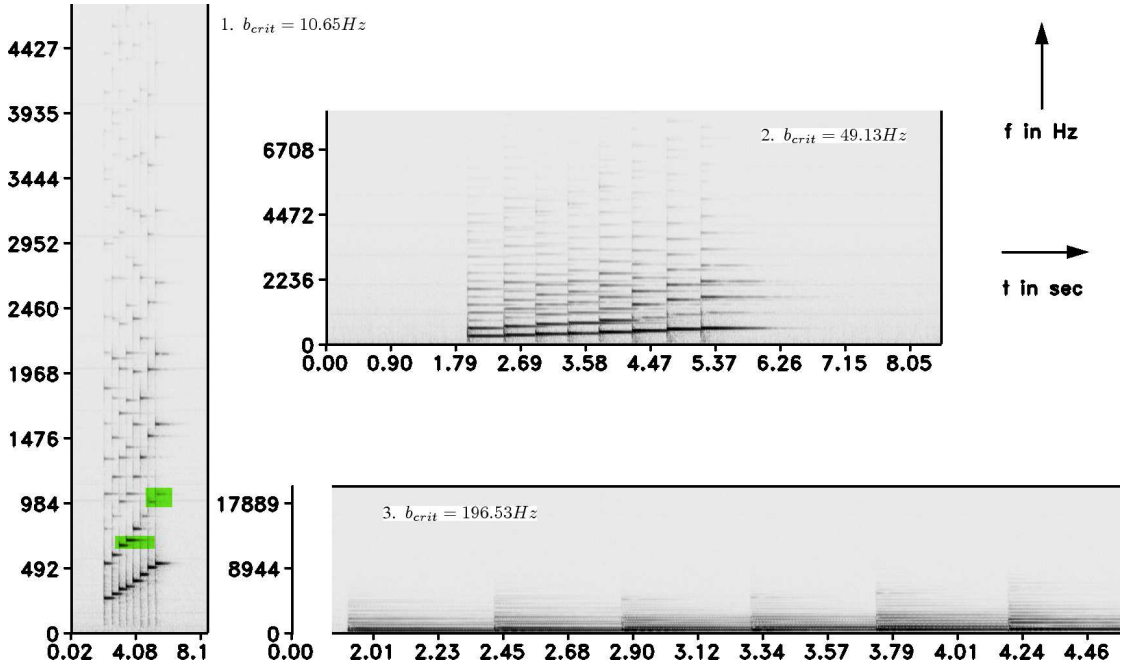


Figure 10: Sound of a piano playing a C-major scale, each note separately. Time-frequency resolutions: 1. $b_{crit} = 10.65Hz$, 2. $b_{crit} = 49.13Hz$ and 3. $b_{crit} = 196.53Hz$.

- combination masks

Rectangle mask: Figure 8 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 11 shows left an example of a click of a ball-pen and right an example of a whistling sound.

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 12 ($b_{crit} = 11.46Hz$) shows again the sound of a piano playing a C-major scale, each note separately (left). The mask useful in this case 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 12 shows the spectrogram of a short piano piece ($b_{crit} = 11.46Hz$), which is of course polyphonic (right). The comb structure of each single note is nevertheless preserved.

Polygon masks: The last mask we want to explain simply uses polygons. See figure 13 for an

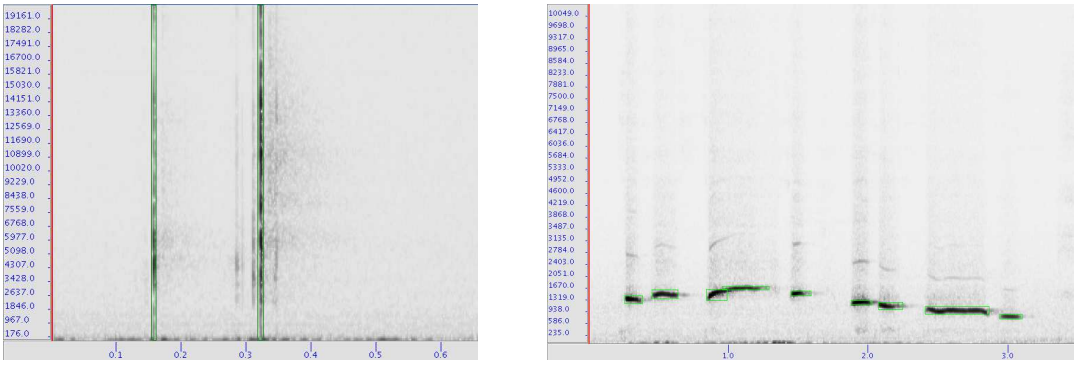


Figure 11: Rectangle mask in the two extreme shapes. Left ($b_{crit} = 196.53Hz$): Click of a ball-pen. Rectangle mask with temporal characteristics. Right ($b_{crit} = 65.51Hz$): whistling, strong tonal character.

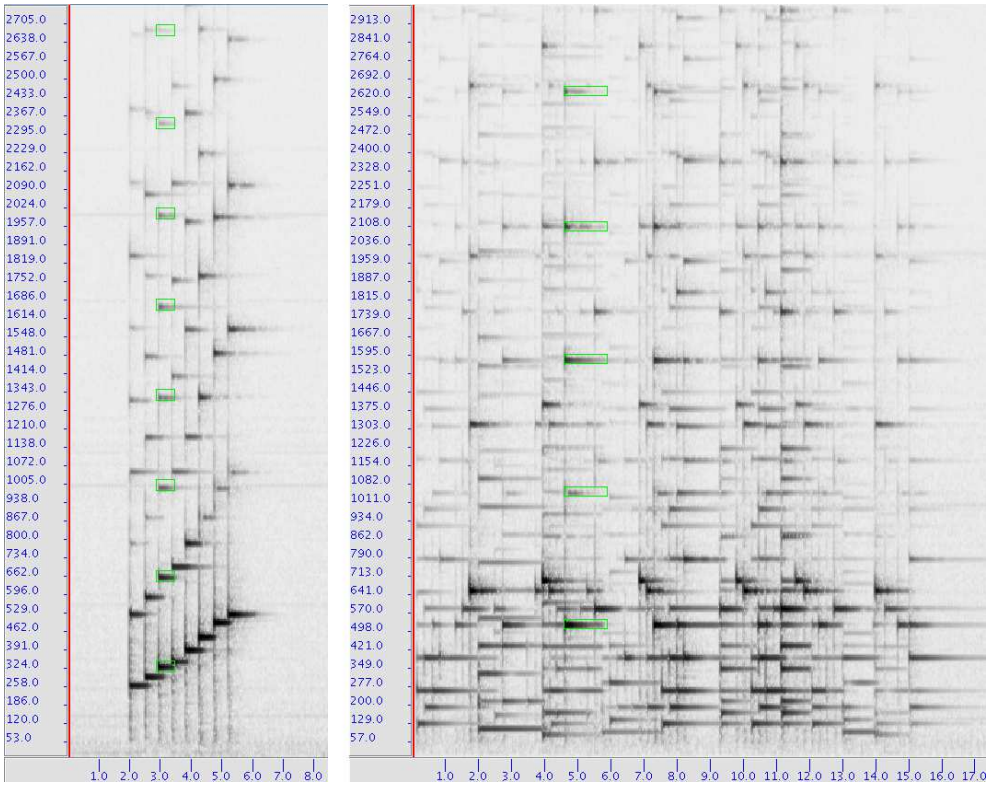


Figure 12: Sound of a piano, $b_{crit} = 11.46Hz$. Left: C-major scale. Right: Short polyphonic piece.

example. It shows a speech signal with background noise ($b_{crit} = 78.61Hz$). 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 of surrounding broadband noise.

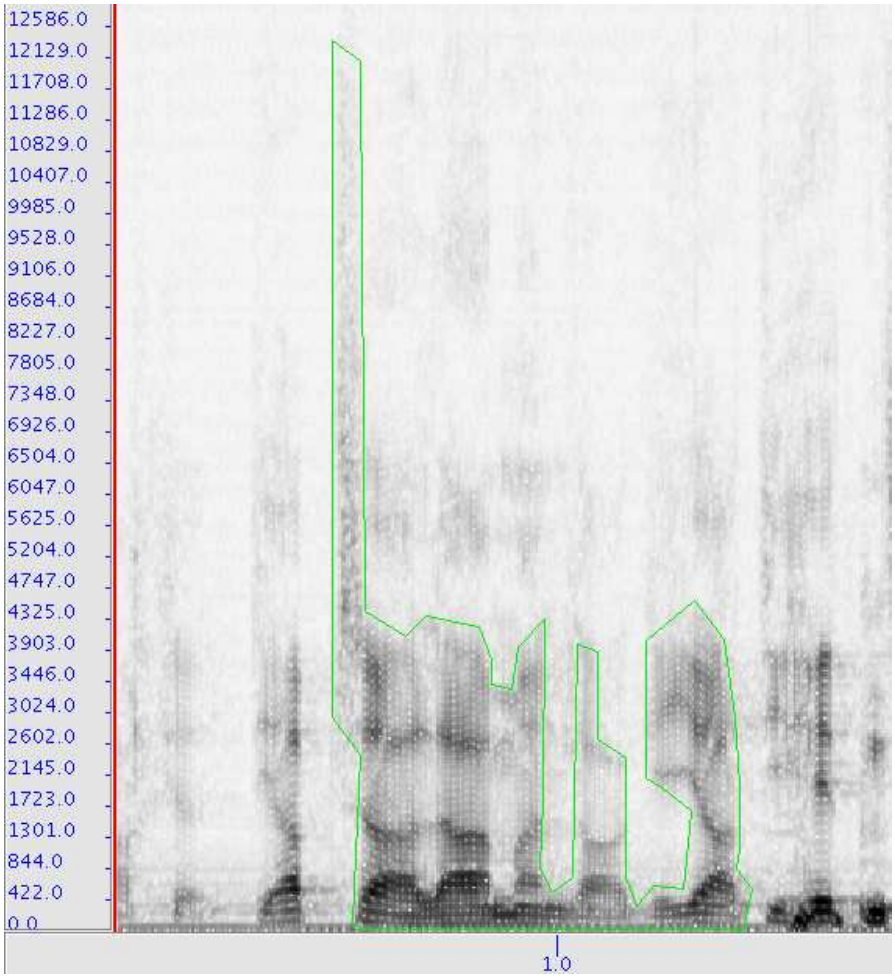


Figure 13: Imaged sound of speech with background noise, $b_{crit} = 78.61Hz$. The polygon selects parts of the speech signal, separating it from the surrounding broadband noise.

Combination masks: To match for complex sounds it is possible to build up complex masks out of repeated 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. Their are several useful possibilities.

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 around in the spectrogram. If only the time position is changed, the same sound is just moved or copied to an other time instance.

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

Changing the shape: If the sound has harmonic components, it is necessary in this case, to stretch or compress the sound in frequency direction, to preserve the harmonic structure. Some more details of this issue are discussed in the next section under the term "image-transformations"

Evaluating: A very helpful mechanism in Visual Audio is the playback of selected regions of an 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 Smart audio brushing

In this section we will discuss smart techniques for Visual Audio Editing, which are based on audio objects. This is a more flexible approach than the one presented in the preceding chapter, because audio objects can adapt to much more complex shapes, than simple masks.

Firstly we will discuss the features of audio objects. A sound recorded beforehand under defined conditions is used as an audio object i.e. as a template mask for editing the current audio track.

Secondly we illustrate the detection of known audio objects. Detections are described by the locations (i.e. positions and shapes) of the audio objects in the spectrograms. Detection has to be performed resilient to typical variations in which audio objects of interest can be experienced.

Last we explain deleting and modifying of detected audio objects. Audio object deletion is a special form of modifying audio objects. An example for a more general modification is editing the audio object in a different track before adding it to the original audio track again. The remaining signal content is at the same time left unchanged. Figure 14 summarizes these two possible two-step editing processes.

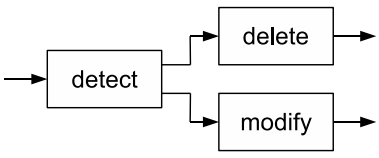


Figure 14: Two-step editing process for audio objects.

4.1 Audio objects: template masks

Reproducible sounds, which are well-structured can be treated as visual objects in the spectrogram. In the spectrogram, they are characterized by a distinctive visual pattern – a pattern which looks similar even under typical variations such as frequency 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 an closing car door or the rattling of a single key on a PC-keyboard. An example is also the click of a ball-pen, whose imaged sound we already discussed in figure 9 and figure 11. An other example is an individual note played by a piano. See for instance figure 15. If the structure of a sound is stable over long times, even

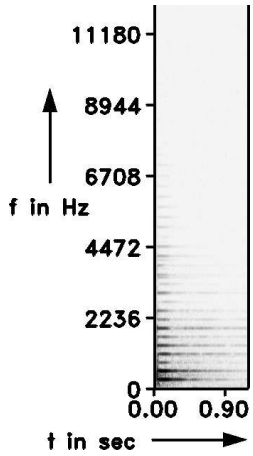


Figure 15: Example for an audio object, which can serve as template masks: A short C2 played as an individual note by a piano. $b_{crit} = 49.13Hz$.

longer portions of a sound can serve as audio-objects. A typical example is a single track of an audio CD.

4.2 Detecting audio objects

Detection starts with a template of an audio object. Figure 16 explains the procedure. The audio

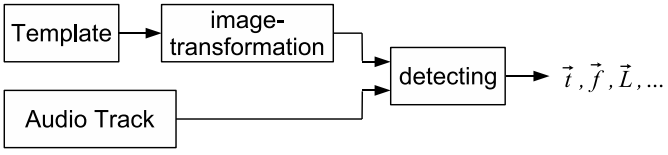


Figure 16: Detecting audio objects.

object template undergoes a set of predefined parametrized image-transformations[4] before a

[4]These transformations, discussed in the image domain, are named image-transformation, to distinguish them from the time-frequency transformations in this chapter.

visual detecting algorithm is used to possibly detect the modified template in the audio track.

Sound variations, image-transformations: An audio object is represented by a visual template. Each template is transformed before comparison with the audio track. This step compensates for the different setting used for recording the template and the audio track (different microphones, different rooms, different instrument of the same kind, etc.). The different settings result in slightly different spectrograms, different levels and small sampling rate differences. While for the first and the second, the detection relies on a robust algorithm, the third can be avoided by the preprocessing step called image-transformation (see Figure 16).

The sampling rate differences have the following impact on the template or an audio track:

- A higher sampling rate results in more samples in a given time. Compared to the correct sampling rate this results in a longer image. The template has to be compressed in time direction to compensate for this.
- A higher sampling rate recording played at the correct sampling rate results in a lower sound. The template has to be stretched in frequency direction to compensate for this.
- Regardless of the sampling rate difference, which is unknown, a compression in one comes along with a stretching in the other direction. The template therefore undergoes a combined stretching and compression.

Detecting: The visual detecting algorithm is used with each possible set of allowed image-transformation parameters in order to find the best matches in the spectrograms under the allowed image-transformation space. As a result, a vector of locations (time, frequency, and shape) and perhaps other parameters such as volume level and alike are given wherever the template has been detected in the audio track.

In our system we use the normalized cross-correlation between the modified audio template image and all possible locations in the spectrogram. As a result, audio object are usually located with pixel accuracy. Since audio editing is very sensitive to inaccuracies, the estimated locations and the associated transformation parameters of the reference (i.e. template) audio object could be further refined by the Lucas-Kanade feature tracker ([14]) – an optical flow method – leading to subpixel accuracy. This technique for example is used in conjunction with sub-pixel accurate feature point localization in camera calibration (see [12]).

4.3 Deleting audio objects

An audio object, which has been detected in the audio track, can now be edited by the use of the best matching template. Possible editing tasks are: correcting the volume level, applying a selected equalization or deleting the sound object. In this section, the two most important methods for deleting audio objects are mentioned.

Stamping: The first approach simply “stamps” out the template out, i.e. the magnitude values are set to zero. Either a user decides interactively, by inspecting the spectrograms visually and the result aurally, or the template is applied automatically. All magnitude values, which in the template spectrogram are larger than a certain frequency dependent threshold value are stamped in the audio track (see figure 17).

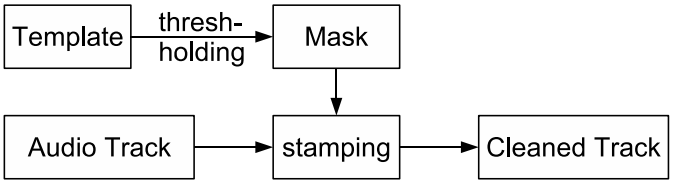


Figure 17: Scheme for stamping a detected audio object with a template spectrogram.

We apply this approach to a mixed music and whistle signal. Figure 18 shows the relevant signals and spectrograms: a) music signal, b) whistling signal, c) mixed signals. Both signals can be recognized in the spectrogram and the time delay of the whistling of $4sec$ can be determined easily. In figure 18, d) the "cleaned" signal is shown, i.e. the whistling is stamped out with the template spectrogram, which was created from a different recording of the same whistle signal. Figure 18 e) shows the over-compensated signal parts and figure 18 f) the under-compensated signal parts. In the cleaned signal the whistling is hardly perceivable and the speech signal has no perceivable difference to the uninterfered original.

Energy based erasing: In contrast to visual objects, which are often intransparent, audio objects are always additive, i.e. they shine through the energy of an other 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 the magnitude values of the templates spectrogram from the magnitude values of the mixed signals spectrogram. As the template was recorded with a different microphone and perhaps has a different level, it is first adapted in order to match the mixed signals spectrogram absolutely and per frequency band as well as possibly before applying the difference. Figure 19 shows a scheme for erasing an audio object by subtracting the energy of an adapted template spectrogram. The success of this method depends on the similarity of template and original signal and the adaption of the template to the original signal.

5 Conclusion

We presented a new approach of editing audio in the spectrogram. 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 automatic. We explained how to adapt the Gabor transformation for this task in order to always reach the absolute physical limit of time-frequency resolution. This is a great advantage, because the ear itself approaches this limit very closely and is very sensitive to errors in audio signals. We presented manual as well as smart user-assisted audio brushing techniques, which are based on the use of known audio objects. Audio objects in general enhance the flexibility in achieving high quality editing results.

Future developments could be to adopt advanced image manipulation techniques, such as inpainting, to reconstruct erased or damaged audio parts and the use of artificial neural nets and other machine learning techniques to further automate the editing processes. Visual Audio will unfold its full power by combining classical techniques with presented techniques in one tool.

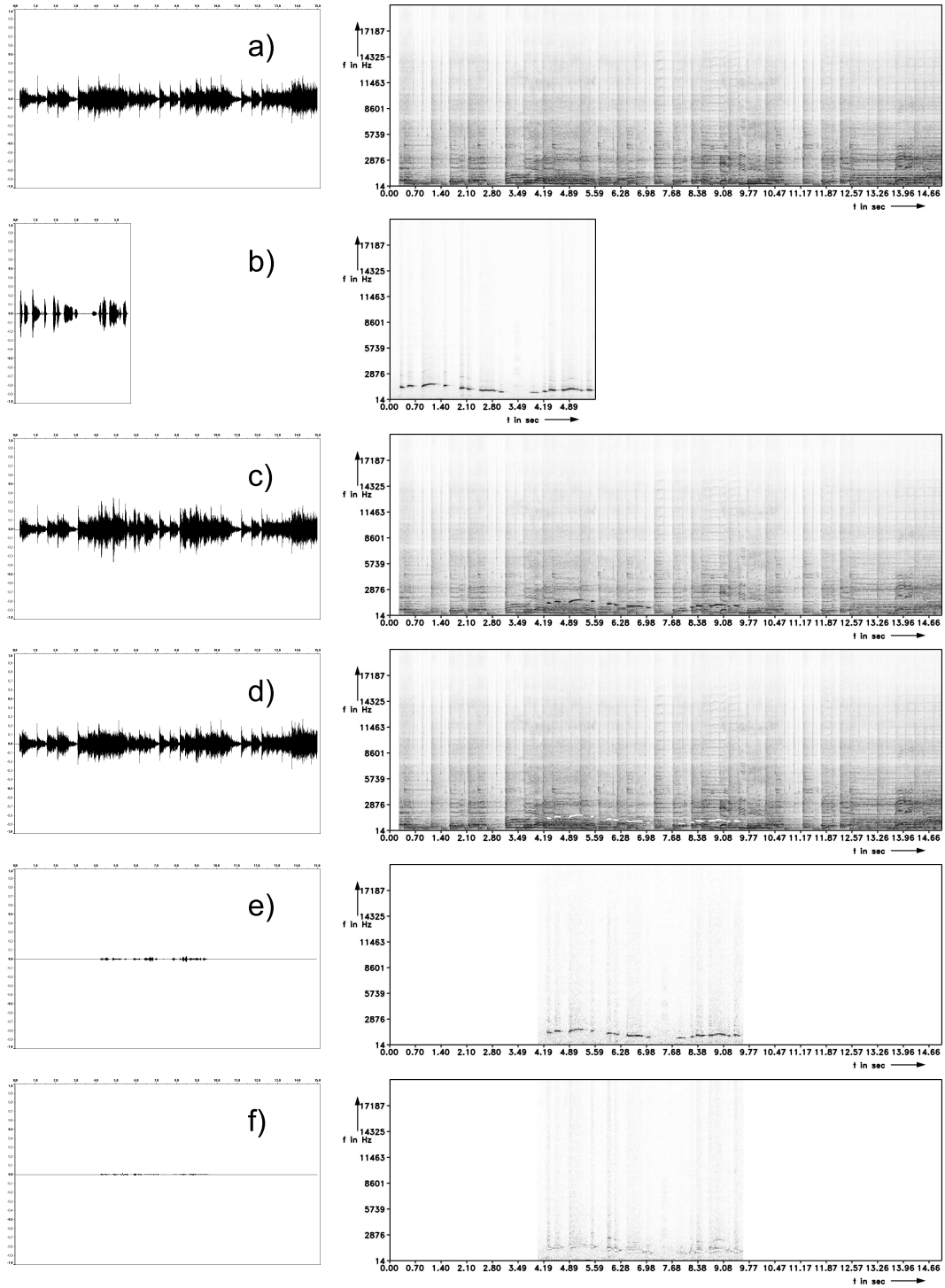


Figure 18: From top to bottom a) music signal, b) whistling signal, c) mixed signals, whistling $4sec$ delayed, d) stamped signal, i.e. cleaned signal, e) over-compensated signal parts and f) under-compensated signal parts.

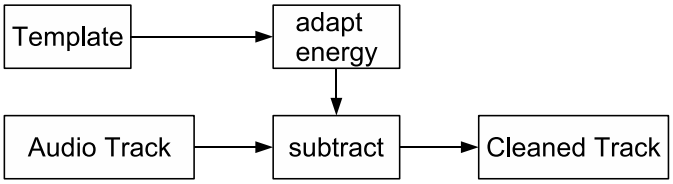


Figure 19: Scheme for erasing a detected audio object with a template spectrogram by subtracting the magnitude values.

References

[1] Martin J. Bastiaans. Optimum sampling distances in the gabor scheme. *Proc. CSSP-97, Mierlo, Netherlands*, pages 35–42, 1997.

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

[3] C. G. v. d. Boogaart and R. Lienhart. Fast gabor transformation for processing high quality audio. Technical Report 2005-21, University of Augsburg, Institute of Computer Science, 2005.

[4] Ingrid Daubechies. *Ten lectures on wavelets*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1992.

[5] D. Donoho. Nonlinear wavelet methods for recovery of signals, densities, and spectra from indirect and noisy data. In *Different Perspectives on Wavelets*, volume 47 of *Proceeding of Symposia in Applied Mathematics*, pages 173–205, 1993.

[6] Hans G. Feichtinger and Thomas Strohmer. *Gabor Analysis and Algorithms: Theory and Applications*. Birkhäuser, Boston, 1998.

[7] Hans G. Feichtinger and Thomas Strohmer. *Advances in Gabor Analysis*. Birkhäuser, Boston, 2003.

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

[9] J. Haitsma and T. Kalker. A highly robust audio fingerprinting system. *Third International Conference on Music Information Retrieval*, pages 107–115, 2002.

[10] J. Hartung, B. Elpelt, and K. Klösener. *Statistik. Lehr- und Handbuch der angewandten Statistik*. Oldenbourg, München, 1995.

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

[12] Eva Hörster, Rainer Lienhart, Wolfgang Kellerman, and J.-Y. Bouguet. Calibration of visual sensors and actuators in distributed computing platforms. *3rd ACM International Workshop on Video Surveillance & Sensor Networks*, November 2005.

[13] Barbara Burke Hubbard. *The World According to Wavelets: The Story of a Mathematical Technique in the Making*. A K Peters, Wellesley, Massachusetts, 1996.

[14] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI '81)*, pages 674–679, April 1981.

[15] Henrique S. Malvar. *Signal Processing with Lapped Transforms*. Artech House, Inc., Norwood, MA, USA, 1992.

[16] Alan V. Oppenheim and Ronald W. Schafer. *Discrete-time signal processing*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989.

[17] S. Qian and D. Chen. *Joint time-frequency analysis: methods and applications*. Prentice Hall, New Jersey, 1996.

[18] D. A. Reynolds. An overview of automatic speaker recognition technology. In *Proc. ICASSP*, volume 4, pages 4072–4075, 2002.

[19] O. Rioul and M. Vetterli. Wavelets and signal processing. *IEEE Signal Processing Magazine*, 8(4):14–38, 1991.

[20] Thomas Strohmer. Approximation of dual gabor frames, window decay, and wireless communications. *Appl.Comp.Harm.Anal.*, 11(2):243–262, 2001.

[21] P. A. Tipler. *Physik*. Spektrum Akademischer Verlag, Heiderlberg, Berlin, 1994.

[22] Patrick J. Wolfe, Simon J. Godsill, and Monika Dörfler. Multi-gabor dictionaries for audio time-frequency analysis. *Proceedings of WASPAA 2001*, 2001.

[23] Ian T. Young, Lucas J. van Vliet, and Michael van Ginkel. Recursive gabor filtering. *IEEE Transactions on Signal Processing*, 50:2798–2805, 2002.

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