Chapter 2
Acoustic Features and Modelling

This chapter gives an overview of the methods for speech and music analysis. The methods described, include all the relevant processing steps from an audio signal to a classification result (Fig. 2.1). These steps include pre-processing and segmentation of the input, feature extraction (i.e., computation of acoustic Low-level Descriptors (LLDs) and summarisation of these descriptors in high level segments), and modelling (e.g., classification).

A particular focus is put in this thesis on real-time processing and the capabilities of the methods to work in systems which require incremental on-line processing of the audio input. An in-depth discussion of real-time, incremental processing is given in Chap. 4.

In this chapter, first, the basic concepts of digital audio signal processing and analysis are described in Sect. 2.1 and common terms are defined. Moreover, the most important pre-processing steps, which are commonly used, are presented. The acoustic LLDs that have been implemented for, and investigated in this thesis are covered in Sect. 2.2. Methods for summarising LLDs over a longer time segment are discussed in Sect. 2.4, and static and dynamic modelling methods are described in Sect. 2.5.

2.1 Basics of Signal Processing

The following provides a brief introduction to signal representation theories and defines some important terms of the area of digital signal processing, which will be used throughout the remainder of this thesis. This introduction is kept very brief on purpose, as the focus of this thesis is neither digital signal processing nor signal representation. It shall only serve the purpose of introducing the most important concepts which are required for understanding the following chapters. For an excellent, deeper discussion of signal representation theories the reader is referred to Oppenheim and Schafer (1975) and Oppenheim et al. (1999).

© Springer International Publishing Switzerland 2016
F. Eyben, *Real-time Speech and Music Classification by Large Audio Feature Space Extraction*, Springer Theses, DOI 10.1007/978-3-319-27299-3_2

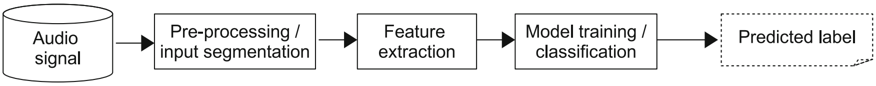
Audio signal
Pre-processing / input segmentation
Feature extraction
Model training / classification
Predicted label

Fig. 2.1 Overview of steps of processing (simplified) for general speech and music analysis methods

2.1.1 Signal Representation

In the following a continuous signal a at time t is defined to have an amplitude of $a(t)$. Such a signal a might be represented by an electrical current in the physical world—e.g., an analogue audio signal, such as the current induced by the vibration of a microphone membrane over a coil. In order to process a signal a in a digital system, the signal must be discretised both in terms of amplitude and time, since processors can represent values only with a finite precision and can only store a finite amount of values.

The discretisation in time is referred to as Nyquist–Shannon sampling (Oppenheim et al. 1996). Thereby the time continuous signal $a(t)$ is represented by a fixed amount of N values $a(n)$ (samples) per unit of time. The sampling rate or sampling frequency f_s is the frequency at which the values $a(n)$ are sampled from the time continuous signal $a(t)$. The relation between the discrete time index $\hat{n}$ and the continuous time t is given by the sampling period T_s

$$T_s = \frac{1}{f_s} \tag{2.1}$$

as:

$$t = n \cdot T_s. \tag{2.2}$$

In order to be able to reconstruct the continuous analogue signal from the finite set of N samples, the following condition for the sampling frequency—called the sampling theorem—has to be met (cf. Oppenheim et al. 1996):

$$f_s \geq 2 f_h, \tag{2.3}$$

where f_h is the highest frequency present in the original signal $a(t)$, also referred to as the Nyquist frequency. If this condition is met, the original time continuous signal can be reconstructed from the sampled signal by low-pass filtering, and the signal $a(n)$ contains all the information from $a(t)$. To ensure the sampling theorem for any type of input, in practice an analogue low-pass filter is applied to $a(t)$ before the signal $a(n)$ is sampled from $a(t)$.

Next, each sample $a(n)$ must be representable by a finite set of values, i.e., using a finite precision. This conversion of a continuous amplitude $a(n)$ to a discrete amplitude $x(n)$ is referred to as quantisation (Gregg 1977). A fixed set of discrete values is defined for the expected range of the signal and each continuous amplitude $a(n)$ is mapped to the nearest discrete value of the defined set, yielding $x(n)$. The difference between the actual value $a(n)$ and the resulting discretised value $x(n)$ introduces a small error, referred to as quantisation error. This error is irreversible and—in contrast to the time discretisation—cannot be eliminated when reconstructing the time and value continuous signal $a(t)$. The quantisation error can be seen as additive noise on top of the original signal $a(t)$ and is thus often referred to as quantisation noise.

On typical digital signal processing platforms a precision of $b = 16$ or $b = 24$ bits is used. The number of possible sample values is given by 2^b, and is 65,536 for 16-bit precision and 16,777,216 for 24-bit precision. The samples can be stored as integer values, or as floating point numbers.

For the analysis of speech and music signals it is important for all values to have a common range of values. Thus, it was decided to represent all sample values as 32-bit floating point values in the ongoing and to scale the values to the range $[-1; +1]$. This ensures a common representation regardless of the precision of the input. The scaling of a sample x_b with b-bits precision to the scaled sample x is performed according to:

$$x = \frac{x_b}{2^{b-1}}. \tag{2.4}$$

Note, that this assumes x_b to be represented by a signed integer type, e.g., a range of -32768 – $+32767$ for 16-bit integers.

2.1.2 Frequency Domain

The signal $a(t)$ can also be represented in the frequency domain by a superposition of sinusoidal base functions. According to Fourier (1822) (cf. also Lejeune-Dirichlet 1829) any band-limited, finite time signal can be represented by finite superposition of sines and cosines with different frequencies (f), magnitudes (A), and phases (ϕ) (=Fourier Series). I.e., $a(t)$ is represented by $A(f)$ and $\phi(f)$ with a finite set of frequencies f. The process of estimating the magnitudes and phases of the base functions is called Fourier Transformation (FT). As the following deals with signals $x(n)$ which are both discrete in time and value, only the Discrete Fourier Transformation (DFT) will be introduced briefly at this point. For more details and a discussion of the Fourier Transformation the interested reader is referred to Oppenheim et al. (1999) and Lizorkin (2002).

The DFT of a real valued signal $x(n)$ with discrete time index $n = \frac{t}{T_s}$ is defined for the integer discrete frequency (bin index) $m = \frac{f}{f_0}$ and $m \in [0; M]$ ($M = \frac{N}{2} + 1$) as:

$$X(m) = \sum_{n=0}^{N-1} x(n) e^{\frac{-j2\pi mn}{N}}. \tag{2.5}$$

For more general considerations in the remainder of this thesis, the bin index m shall be converted to a linear frequency f via a function $F(m)$ by Eq. (2.6):

$$f = F(m) \tag{2.6}$$
$$\text{and } m = F^{-1}(f). \tag{2.7}$$

In the linear case of the DFT, the above is:

$$F_{lin,Hz}(m) = mf_0 \tag{2.8}$$
$$F^{-1}_{lin,Hz}(f) = \frac{f}{f_0}. \tag{2.9}$$

The result $X(m)$ is a complex value (j indicates imaginary parts in Eq. (2.5)) which gives both the magnitude and the phase of the corresponding mth base function. The DFT base functions $e^{\frac{-j2\pi mn}{N}}$ are orthogonal if m is chosen as an integer value, i.e., the DFT is computed only for multiples of a base frequency f_0. For the DFT f_0 is given in terms of the sampling frequency T_s and the frame size N as follows:

$$f_0 = \frac{1}{N \cdot T_s}. \tag{2.10}$$

The number of discrete frequency bins M is then given as $M = \frac{N}{2}$. For $m = 0$ the direct current (DC) component of $x(n)$ is returned, while for $m = M$ the magnitude of the Nyquist frequency is returned. The computation of a DFT has an asymptotic complexity of $\mathcal{O}(N^2)$. In practice, an optimised algorithm is implemented: the Fast Fourier Transformation (FFT) (Cooley et al. 1969). The algorithm uses the principle of Divide and Conquer and splits the DFT in two sub-problems of half the size of the original problem. The FFT achieves an asymptotic complexity of $\mathcal{O}(N \log(N))$ but requires the frame size N to be a power of two. If a frame with a size which is not a power of two needs to be transformed, the typical procedure is to apply zero-padding to the frame, i.e., the frame size is increased to the next higher power of two and the additional samples are filled with zeros.[1]

For the human ear only the magnitudes of the components $X(m)$ are relevant, and not the phase. The phase is only necessary in very special analysis applications which require instantaneous frequency estimates, for example, or for a proper reconstruction of $x(n)$ from $X(m)$. For analysis, each component $X(m)$ is therefore converted to the magnitude $X_M(m)$:

$$X_M(m) = |X(m)| = \sqrt{Im(X(m))^2 + Re(X(m))^2}. \tag{2.11}$$

[1] In openSMILE the FFT with complex valued output (and also the inverse FFT) is implemented by the `cTransformFFT` component. Magnitude and Phase can be computed with the `cFFTmagphase` component.

The phase $X_\phi(m)$ in radians is given as:

$$X_\phi(m) = arctan\left(\frac{Im(X(m))}{Re(X(m))}\right). \tag{2.12}$$

The signal $x(n)$ can also be resynthesised from the spectral magnitudes and phases. Even though this is not required for analysis of speech and music directly, it might be required indirectly, e.g., if a filter is to be implemented in the spectral domain and time domain descriptors are to be computed from the resulting signal. Also, in this way, various audio features (cf. Sect. 2.2) can be transformed into an audio signal, i.e., feature trajectories can be made audible.

$X(m)$ is obtained from a magnitude/phase representation as:

$$X_m(m) = X_M(m)e^{jX_\phi(m)} = X_M(m)\left(\cos(X_\phi(m)) + j\sin(X_\phi(m))\right). \tag{2.13}$$

The time domain signal $x(n)$ is obtained with the inverse real-valued DFT:

$$x(n) = \frac{2}{M}\sum_{m=-M}^{M} X(m)e^{\frac{j2\pi mn}{N}}. \tag{2.14}$$

2.1.3 *Short-Time Analysis*

For audio analysis tasks the spectrum contains important information, including very obvious attributes such as information about pitches of musical instruments, or the pitch of a speaker. The time domain signal contains information about amplitude, etc. However, all these attributes change over time and we need to find a way to estimate these attributes periodically, in (quasi-)stationary segments, instead of performing a single global analysis over the whole signal $x(n)$ or the spectrum $X(m)$ computed from the whole signal $x(n)$.

The Concept of Windowing

To solve this problem, commonly the method of short-time analysis (also referred to as framing or windowing) is considered.[2] Thereby a signal $x(n)$ with $n \in [0, N[$ is divided into K short, overlapping frames $x_k(\hat{n})$ ($k \in [0 \ldots K-1]$) of N_f samples or $L_f = N_f \cdot T_s$ seconds length. The discrete time index $\hat{n}$ within each frame is measured relative to the start of the frame, i.e., $\hat{n} \in [0 \ldots N_f - 1]$. The start index $n_{start,k}$ in the signal $x(n)$ which corresponds to the start of the kth frame is given as:

$$n_{start,k} = k \cdot N_f^{(T)}, \tag{2.15}$$

[2] In openSMILE windowing of audio samples (i.e., short-time analysis) can be performed with the `cFramer` component.

where $N_f^{(T)} = \frac{T_f}{T_s}$ is the frame period measured in samples and T_f is the frame period measured in seconds, i.e., the time between the start of two consecutive frames. The names frame step, frame shift, frame period, and frame increment all refer to this same quantity, measured either in seconds or samples.

Typically T_f is smaller than L_f because overlapping frames are used. The percentage of overlap O_f between adjacent frames is defined as:

$$O_f = \frac{L_f - T_f}{L_f}. \tag{2.16}$$

The end index $n_{end,k}$ of the frame is given as:

$$n_{end,k} = k \cdot N_f^{(T)} + N_f. \tag{2.17}$$

For the start and end times of the kth frame relative to the signal $x(t)$ holds:

$$t_{start,k} = n_{start,k} \cdot T_s = k \cdot T_{f,s} \cdot T_s = k \cdot T_f \tag{2.18}$$

$$t_{end,k} = n_{end,k} \cdot T_s = (k \cdot T_{f,s} + N_{f,s}) \cdot T_s = k \cdot T_f + N_f \tag{2.19}$$

In addition to the theoretical framework of short-time analysis laid out above, in practice border conditions have to be addressed. That is in particular, how to deal with N_e excess samples at the end of the signal $x(n)$ for which $N_e < N_f$. Basically three strategies exist:

1. Ignoring these N_e samples,
2. Dealing with a smaller frame at the end, which contains the last N_e samples.
3. Appending $N_f - N_e$ samples to the end of the signal $x(n)$.

In this thesis, strategy (1) is implemented, as the analysed segments are assumed to be much longer than a single frame. It thus can be safely assumed that these discarded samples at the end will not contribute much to the final result. If we choose the frame length L_f to be sufficiently small, we can assume the signal's properties (such as pitch of a voice or an instrument) to vary only minimally or remain constant within a single frame k, i.e., the signal is assumed to be quasi stationary. Typical frame lengths in speech and music analysis range from 20 milliseconds (ms) to 60 ms. The most commonly chosen frame period is 10 ms, which originates from the domain of Automatic Speech Recognition (ASR) (Rabiner 1989; Young et al. 2006). In some applications where a higher temporal accuracy is required, also 5 ms frame period is considered; also 20 ms frame periods are used and are feasible in some analysis tasks.

The concept of framing laid out above is not limited to short-time analysis of audio samples. Any time-value series can be segmented into 'short' time segments. For instance, parameters extracted from short audio frames, such as signal energy (see Sect. 2.2.2), can also be viewed as a time-value series with a sampling period of e.g., 10 ms. In the ongoing, the terms 'frame' and 'window' refer to a frame of audio samples, if not explicitly specified otherwise. The term ‘segment’ refers to a higher level segment, e.g., one second of signal energy values, or the whole signal. An in depth discussion of such higher level segment features is contained in Sect. 2.4 and a discussion of the choice of the segment length is found in Sect. 4.1.

Note: To simplify the equations in the ongoing and foster generalisation, it will not be discriminated between frames $x^{(k)}(\hat{n})$ and a signal $x(n)$ on the symbolic level, unless explicitly necessary. The symbol $x(n)$ will be used for a general signal of length N in the equations. An optional discrete time index k as superscript $x^{(k)}(n)$ will denote that this signal represents or is derived from a frame at a discrete frame index k. The text will further clarify on whether frames, higher level segments, or the whole signal is considered.

Window Functions

The framing described in the previous section corresponds to a multiplication of the signal $x(n)$ with a rectangular window function $w_r(n)$ for the kth frame x_k as follows:

$$x_k(\hat{n}) = x(k \cdot N_f^{(T)} + \hat{n}) \cdot w_r(\hat{n}). \tag{2.20}$$

The rectangular window function is defined as:

$$w_r(n) = 1 \text{ for } n = 0 \ldots N_f \tag{2.21}$$

In the spectral domain (cf. Sect. 2.1.2) such a multiplication corresponds to a convolution. Therefore, the ideal windowing function—i.e., one which does not distort the spectrum—has a dirac impulse shaped spectrum. In the time domain this corresponds to a constant function of infinite duration. Since the window always has a finite duration, a certain amount of spectral distortion due to the framing can never be avoided, regardless of the choice of the windowing function. In fact, a compromise between time and frequency domain properties of the windowing function must be found. In the time domain a finite duration with steep edges is preferred (rectangle like), while in the frequency domain a very narrow main maximum (dirac like) is preferred with near zero side maxima.

In the following a discussion of the most common windowing functions (as they are implemented in openSMILE[3]) and their properties is contained:

Rectangular window The Rectangular Window is defined as a constant function for $n \in [0, N-1]$: $w_{Rec}(n) = 1$. It corresponds to a *sinc*-function in the spectral domain $W(m)$:

$$W_{Rec}(m) = \frac{\sin(m)}{m} = sinc(m) \tag{2.22}$$

[3] http://opensmile.audeering.com.

The Rectangular Window is best suited for analysis in the time domain such as for Zero-Crossing Rate or Amplitude descriptors. It is very efficient to compute, as no multiplications are required. Due to the high side maxima of the sinc function it is not recommended for frequency domain analysis.

Hann(ing) window The Hanning window—also known as Hann-window or the raised cosine window—is named after the Austrian meteorologist Julius von Hann by Blackman and Tukey (1959). It is defined for $n \in [0, N-1]$ as:

$$w_{Han}(n) = 0.5\left(1 - \cos\left(\frac{2\pi n}{N-1}\right)\right) \tag{2.23}$$

The side lobes in the spectrum roll off by approximately 18 dB per octave, which makes this window suitable for spectral analysis. Moreover, the symmetry in the time domain and the fact that the amplitude reaches zero at both sides of the window makes this window perfectly suitable for applications in which $x(n)$ has to be reconstructed from spectra with the overlap-add method when the overlap of the frames is 50 % (Oppenheim and Schafer 1975).

Hamming window A modification of the Hanning window is the Hamming window. It reduces the amplitude of the first sidelobe in the spectrum significantly (by about one fith), at the cost of higher amplitudes for the higher order sidelobes (Enochson and Otnes 1968). In contrast to the Hanning window, it does not reach zero amplitude at the sides. It is the most commonly used window for analysis in the spectral domain, especially in speech recognition and speech analysis. There, according to Young et al. (2006), it is defined as follows for $n \in [0, N-1]$:

$$w_{Ham}(n) = \alpha - \beta \cos\left(\frac{2\pi n}{N-1}\right). \tag{2.24}$$

with $\alpha = 0.54$ and $\beta = 0.46$. A more in depth discussion, as well as a more precisie definition of the coefficients is found in (Harris 1978). According to Harris, the theoretical optimal values for the coefficients are $\alpha = \frac{25}{46}$ and $\beta = \frac{21}{46}$. Later, Nuttal (1981) proposed the following values as optimal with respect to minimizing the sidelobes in the spectrum: $\alpha = 0.53836$ and $\beta = 0.46164$. In the automatic speech recognition community, however, the definition of Young et al. (2006) is widely used. Therefore, only this definition is used in this thesis.

Gaussian window The Gaussian window uses a Gaussian function as windowing function ($n \in [0, N-1]$):

$$w_{Gau}(n) = e^{-\frac{1}{2}\left(\frac{n-(N-1)/2}{\sigma(N-1)/2}\right)^2} \tag{2.25}$$

The Gaussian function has the special property of being an eigenfunction of the Fourier Transform, i.e., when—transformed to the spectral domain—it remains Gaussian shaped (Oppenheim et al. 1999). It therefore has no sidelobes; the bandwidth (defined by the standard deviation of the Gaussian in the spectral domain) is inversely proportional to standard deviation of the Gaussian in the time domain. In the time domain it does not touch zero at the ends of the window due to the infinite length of the Gaussian function. In some applications it might be necessary to have zeroes at the ends of the window. This can be achieved by multiplying the Gaussian window with another window, e.g., a Hanning window.

Sine/Cosine window In contrast to a raised cosine (Hanning) window, the sine window consists of the first sine half-wave ($n \in [0, N-1]$) (Oppenheim et al. 1999):

$$w_{Sin}(n) = \sin\left(\frac{\pi n}{N-1}\right) \tag{2.26}$$

Due to its steepness at the window ends it has large side maxima in the spectrum and thus is less preferred.

Triangular window For some applications where overlapping windows need to be resynthesised in the time domain, triangular windows might be considered due to their symmetry. However, they have less favorable spectral properties than, e.g., a Hann window. A general Triangular window with non zero-valued end points is defined as (for $n \in [0, N-1]$) (Oppenheim et al. 1999):

$$w_{Tri}(n) = \begin{cases} \frac{2(n+1)}{N} & \text{if } n < \frac{N}{2} \\ \frac{2(N-n)}{N} & \text{if } n \geq \frac{N}{2} \end{cases} . \tag{2.27}$$

Bartlett window If the window must have zeroes at the endpoints, a Bartlett window can be used. The Bartlett window is a Triangular window with zero-valued end points defined as (for $n \in [0, N-1]$) (Oppenheim et al. 1999):

$$w_{Bar}(n) = \begin{cases} \frac{2n}{N-1} & \text{if } n < \frac{N}{2} \\ \frac{2(N-n-1)}{N-1} & \text{if } n \geq \frac{N}{2} \end{cases} . \tag{2.28}$$

A triangular window with zero valued endpoints can be expressed by the convolution of two rectangular windows in the time domain. Thus, the resulting spectral shape is that of a squared $sinc(m)$ function: $sinc^2(m) = \left(\frac{sin(m)}{m}\right)^2$ function.

Lanczos window For the completeness of this overview on windowing functions, the following contains a list of further functions without discussion. These functions are used for specific digital signal processing applications, but are only of minor importance for analysis of speech and music signals.

The Lanczos window represents a sinc function in the time domain, trimmed to a finite length $n \in [0, N-1]$, which includes only the main maximum:

$$w_{Lac}(n) = \frac{\sin\left(\pi \frac{2n}{N-1} - 1\right)}{\frac{2n}{N-1} - 1} \tag{2.29}$$

It was introduced for the purpose of Lanczos resampling (Turkowski and Gabriel 1990).

Blackmann window A Blackmann window is defined for $n \in [0, N-1]$ as:

$$w_{Bla}(n) = \frac{1-\alpha}{2} - \frac{1}{2}\cos\left(\frac{2\pi n}{N-1}\right) + \frac{\alpha}{2}\cos\left(\frac{4\pi n}{N-1}\right) \tag{2.30}$$

The suggested default for the α parameter is 0.16. It is based on the Blackman Function (Blackman and Tukey 1959).

2.1.4 *Pre-processing*

Before audio features (see Sect. 2.2) are extracted from speech or music signals, several pre-processing steps can be applied to the signal. In general, pre-processing refers to everything done to the signal (in time or frequency domain) before LLDs are extracted. The typical pre-processing for speech and music analysis, however, is limited to the time domain and concerns the following steps:

Down-mixing, i.e., conversion of multi-channel signals to a single channel (mono). Instead of extracting near redundant features for both channels individually, most often a stereophonic signal is converted to a monophonic signal prior to feature extraction. The most simple procedure is linear down-mixing, where the samples of all C channels are linearly averaged to a single channel x_0:

$$x_0(n) = \frac{1}{C}\sum_{c=1}^{C} x_c(n). \tag{2.31}$$

Down-mixing is not performed when multi-channel features are to be extracted, such as spatial features, or when reconstructing a single source signal from a multi-channel recording (source separation). Both aspects are, however, not in the scope of this thesis, thus monophonic down-mixing is always applied here.

Re-sampling, i.e., change of the sampling frequency to a common value for inputs with different sampling frequencies, or reduction of the sampling frequency in order to speed up the analysis. In speech, the relevant frequency range is from ≈50 Hz to ≈6 kHz, and in music from ≈50 Hz to ≈8 kHz. A sampling frequency of 16 kHz is thus sufficient for analysis of most speech and music signals.

Pre-emphasis,[4] i.e., filtering of the signal to attenuate frequency bands which carry important information. For speech analysis typically a 1st order high-pass filter is applied to a signal $x(n)$ in order to emphasise information on formants (Young et al. 2006), yielding the pre-emphasised signal $x_p(n)$:

$$x_p(n) = x(n) - kx(n-1), \tag{2.32}$$

where $k \in [0; 1]$ is the pre-emphasis coefficient, which controls the strength of the pre-emphasis (1 is most, 0 is least). Typical values for speech processing range from 0.9 to 0.97 Young et al. (2006). In music processing a band pass filter could be used to emphasise certain octaves, or reduce sub-bass effects, for example. Also, the above 1st order high-pass pre-emphasis filter can be inverted to a low-pass "de-emphasis" filter as follows:

$$x_d(n) = x(n) + k_d x(n-1). \tag{2.33}$$

Other pre-processing steps include noise-reduction and echo cancellation. Some of these methods operate in the spectral domain. As the focus of this thesis is on large scale feature extraction and not on signal enhancement and pre-processing, the reader is referred to (Schuller 2013) for further reading on these topics. All acoustic descriptors described in this thesis can be extracted from enhanced and filtered signals, too.

2.2 Acoustic Low-Level Descriptors

In this section all acoustic LLDs which have been implemented for and evaluated in the course of this thesis are described in detail.

An acoustic LLD is defined as a parameter computed from a short time frame $x_k(n)$ (length N_f) from an audio signal at time $t = k \cdot T_f$. The length of the frame should be chosen in a way to a) ensure quasi stationarity of the signal within the frame with respect to the LLD of interest, and b) ensure that the frame contains enough data to compute the LLD (cf. Sect. 2.1.3 on details of the framing). For some LLDs a window function is applied to the frame prior to computing the LLD (cf. Sect. 2.1.3 for a discussion of windowing functions). Typical frame lengths are 25–32 ms for most LLDs.

[4] In openSMILE pre-emphasis can be implemented with the `cPreemphasis` component on a continuous signal, or with the `cVectorPreemphasis` component on a frame base (Hidden Markov Toolkit (Young et al. 2006) (HTK) compatible behaviour).

2.2.1 *Time Domain Descriptors*

Time domain descriptors are computed directly from the time domain signal $x(n)$. Commonly these include, the number of zero-crossings (Sect. 2.2.1.1), amplitude statistics (Sect. 2.2.1.2), and a DC offset. Strictly speaking, other descriptors such as the signal energy (Sect. 2.2.2), or linear predictive coding coefficients (Sect. 2.2.7.1) are also extracted from the time domain signal. However, they can also be extracted from frequency domain representations of the signal and moreover are more related to spectral characteristics of the signal. Thus, they are not considered in this section.

2.2.1.1 Zero- and Mean-Crossing Rate

The Zero-Crossing Rate (ZCR) describes the number of sign changes c of $x(n)$ per unit of time (usually one second) (Chen 1988):

$$ZCR = \frac{c}{1.0\,\mathrm{s}}. \tag{2.34}$$

A sign change is defined to occur when:

$$x(n-1)x(n) < 0 \tag{2.35}$$

or

$$x(n-1)x(n+1) < 0 \quad \text{and} \quad x(n) = 0. \tag{2.36}$$

In analogy to the ZCR, one can define the Mean-Crossing Rate (MCR) as the rate of changes from below to above the mean μ_x of $x(n)$ (or vice versa). To compute the MCR μ_x is subtracted from $x(n)$ resulting in the mean normalised signal $\hat{x}_n$:

$$\hat{x}(n) = x(n) - \mu_x. \tag{2.37}$$

with

$$\mu_x = \frac{1}{N}\sum_{0}^{N-1} x(n) \tag{2.38}$$

The MCR is then computed from $\hat{x}(n)$ using the same algortihm as for ZCR.

A high ZCR or MCR indicates a signal with much high frequency content. Typically harmonic signals have a low zero crossing rate, which is related to the fundamental frequency of the signal. E.g., a single pure sine will have a zero crossing rate of twice its frequency. White Gaussian noise on the other hand will have a rather high zero crossing rate, due to the broadband high-frequency content. ZCR is used to distinguish voiced speech from unvoiced speech (Bachu et al. 2010) as well as percussive from harmonic parts in music (Gouyon et al. 2000). It is, however, also strongly affected by additive noise, especially at low SNRs. Better metrics have thus been proposed for the voiced/unvoiced decision, as will be shown in the ongoing.

2.2.1.2 Amplitude

Other time domain signal descriptors are the maximum and minimum signal amplitudes, or the maximum absolute value of the amplitudes.

Usually the amplitudes of audio signals are symmetric around 0 amplitude, i.e., the range of the amplitude is from $-a$ to $+a$, and the amplitude is 0 if there is no signal at the input. Sometimes, however, an offset is present, due to various effects, such as a electrical DC offset due to faulty or cheap recording equipment, or—when framing—the influence of extremely low frequencies (which have a period larger than the frame length). In these cases the DC offset of the signal can provide information of intereset. The DC offset of $x(n)$ is equivalent to the mean μ_x of $x(n)$ (cf. Eq. (2.38)).

2.2.2 *Energy*

One of the most basic, yet powerful audio descriptors is the signal energy[5] (Oppenheim et al. 1999). If we assume an audio signal to have no DC offset, i.e., a mean value of zero, the signal energy E for a signal $x(n)$ with $n \in [0; N-1]$, is defined as the sum of squared amplitudes of the signal x:

$$E = \sum_{n=0}^{N-1} x^2(n). \tag{2.39}$$

Often the normalised signal energy is used in order to eliminate the influence of the frame length on the descriptor:

$$E_n = \frac{1}{N} \sum_{n=0}^{N-1} x^2(n). \tag{2.40}$$

In speech and music processing two variations of the signal energy are commonly employed. The first is the Root Mean Square (RMS) energy (Kenney and Keeping 1962):

$$E_{rms} = \sqrt{\frac{1}{N} \sum_{n=0}^{N-1} x^2(n).} \tag{2.41}$$

[5] RMS and logarithmic energy can be computed in openSMILE with the `cEnergy` component.

The second is the logarithmic (log) energy (Young et al. 2006):

$$E_{log} = E_{bias} + E_0 \cdot \log \sum_{n=0}^{N-1} x^2(n), \tag{2.42}$$

where log represents the natural logarithm. E_0 is a scaling factor which is used to scale the logarithmic energy to different unit scales. If $E_0 = 1$, then E_{log} is measured in 'neper', a unit similar to Decibel (dB), only that the natural logarithm is used as a basis. If $E_0 = \frac{10}{\log 10} \approx 4.343$, then E_{log} is measured in dB. For the logarithmic energy it is common to define a 'floor' value, i.e., a minimum value, to avoid very high negative values with high variation due to noise in low energy and silent frames.[6]

The above definitions of the signal energy do not consider any properties of the human hearing and/or human perception of loudness. According to Zwicker and Fastl (1999) it is important to consider human perception for many tasks. An example for identification of stressed syllables in speech is shown where signal amplitude is compared to a loudness measure obtained from a psychoacoustic model. Computing perceptual loudness from a psychoacoustic model is computationally demanding. An approximation $E_{l,approx}$ for the loudness[7] E_l for a narrow-band signal has been used by (Kießling 1997, pp. 156–157):

$$E_{l,approx} = \left(\frac{I}{I_0}\right)^{0.3}, \tag{2.43}$$

where I is the signal intensity defined as the signal energy E of $x(n)$ where $x(n)$ has been weighted with a Hamming window function (cf. Sect. 2.1.3) and I_0 is the reference intensity. For a maximum absolute signal amplitude $|x(n)| = 1.0$ and a reference signal at 60 dB Sound Pressure Level (SPL), $I_0 = 10^{-6}$ is defined (Kießling 1997, pp. 156–157).

The exact measurement of loudness according to a simplified psychoacoustic model is discussed in Sect. 2.2.9.3.

2.2.3 Spectrum

This section introduces various types of magnitude spectra, which can be used as LLDs directly and/or serve as the basis for other descriptors which are computed from the spectra, such as spectral statistics (Sect. 2.2.4) or Cepstral features (Sect. 2.2.10).

[6] openSMILE defines 8.674676×10^{-19} as a floor value for the argument of the log, for samples scaled to the range of -1–$+1$. In case of sample value range from -32767 to $+32767$ (HTK compatible mode), the floor value for the argument of the log is 1.

[7] The loudness approximation and the signal intensity as defined here can be extracted in openSMILE with the `cIntensity` component.

2.2.3.1 Linear Magnitude Spectrum

The spectrum magnitudes $X_M(m)$, as introduced in Sect. 2.1.2 and computed from a short-time frame, can be used as LLDs directly. This spectrum is both linear in frequency m and magnitude X.

In order to make this LLD independent of the analysis frame size N_f (e.g., if inputs with varying sampling rates are analysed), the magnitudes $X_M(m)$ can be normalised by N_f:

$$X_{M,norm}(m) = \frac{X_M(m)}{N_f}. \tag{2.44}$$

The resulting measure $X_{M,norm}(m)$ is the spectral magnitude density. *Note:* in practice, a scaling by the relative energy of the windowing function must be performed when calibrated magnitude/power measurements are required. As this scaling is a constant, however, it can be neglected for feature extraction or speech/music analysis purposes.

The bins $X_M(m)$ represent a vector of basic acoustic descriptors, which contains almost all relevant information from the original signal and—if suitable phase information was available—would allow for reconstruction of the original signal in the time domain. However, raw spectra are not ideal as LLDs because they contain high amounts of redundancy, i.e., individual descriptors (bins, bands, etc.) are highly correlated with each other but are seldom highly correlated to analysis tasks' targets. Therefore, descriptors derived from the raw spectra are preferred, such as spectral statistics (Sect. 2.2.4).

2.2.3.2 Non-linear Magnitude Scales

The linear values $X_M(m)$ or $X_{M,norm}(m)$ correspond to the physical unit of voltage, as they are computed from signal amplitude values. However, human auditory perception is highly non-linear (Zwicker and Fastl 1999) and thus a non-linear representation of the magnitudes might be better suited.

The first step towards human perception is to use power spectra $X_P(m)$:

$$X_P(m) = X_M(m)^2 \tag{2.45}$$

which—despite the name *power* spectra—represent the quadratic energy in each bin, or power spectral densities, accordingly:

$$X_{P,norm}(m) = \frac{|X_{M,norm}(m)|^2}{N_f}, \tag{2.46}$$

which in fact represent the power in each bin, because of the normalisation with the window length.

The next step is the application of a non-linearity. Commonly, a logarithm is applied as non-linearity, or an exponent of $\approx$0.3 is used as non-linearity (e.g., as for the approximation of loudness in Sect. 2.2.2 or for the auditory spectra in Sect. 2.2.9.3).

The spectral magnitudes are converted to a logarithmic spectral power density representation $X_{dBpsd}(m)$ in dB by the following equation[8] (cf. Spanias et al. 2007):

$$X_{dBpsd}(m) = X^{(0)}_{dBpsd} + 20 \cdot \log_{10} \left(|X_{M,norm}(m)|\right) \tag{2.47}$$

Thereby $X^{(0)}_{dBpsd}$ is a normalisation factor for the logarithmic scale and suggested as 90.302 dB by the psychoacoustic model layer I, defined in the MPEG-1 standard.

2.2.3.3 Non-linear Frequency Scales

The magnitudes $X_M(m)$ are equidistant (factor f_0) on a linear (Hz) frequency scale. Human hearing is also non-linear in terms of frequency perception and a non-linear frequency scale has to be preferred (Zwicker and Fastl 1999). This fact has been successfully exploited in the fields of Automatic Speech Recognition, Speaker Recognition, and Music Information Retrieval for a long time by use of Mel-Frequency Cepstral Coefficient (MFCC) features (Rabiner and Juang 1993). Thereby the spectrum is transferred to a so called Mel-Frequency scale (cf. below) before further processing. More details on MFCC are given in Sect. 2.2.10.1 and Young et al. (2006) as well as in Rabiner and Juang (1993).

This section introduces various non-linear frequency scales and discusses methods for transformation of a linear frequency scale spectrum to a non-linear frequency scale by interpolation.[9] Each frequency scale is defined by a forward transformation function $\Theta_{scale}(f)$ of the linear frequency $f^{(lin)}$ (in Hz) to the non-linear frequency $f^{(scale)}$ in units of the respective scale (e.g., Mel or Bark (critical band rate, cf. below)):

$$f^{(scale)} = \Theta_{scale}(f^{(lin)}) \tag{2.48}$$

For some scales also a backward transformation function Θ^{-1}_{scale} is given analytically:

$$f^{(lin)} = \Theta^{-1}_{scale}(f^{(scale)}). \tag{2.49}$$

A discrete magnitude spectrum given for *scale* is denoted by the symbol $X^{(scale)}_M(m^{(scale)})$ where $m^{(scale)}$ is the integer bin index on the non-linear frequency scale which is mapped to $f^{(scale)}$ by a general function (cf. Sect. 2.1.2) $m^{(scale)} = F^{-1^{(scale)}}(f^{(scale)})$.

[8] In openSMILE the option `dBpsd` must be enabled in the `cFftMagphase` component in order to compute logarithmic power spectral densities.

[9] In openSMILE these spectral scale transformations and spline interpolation can be applied with the `cSpecScale` component.

For the transformation of the M linear frequency bins m to $M^{(scale)}$ non-linear frequency bins $m^{(scale)} \in 0 \ldots M^{(scale)}$, first the $M^{(scale)}$ frequencies $f^{(scale)}$ for all the $m^{(scale)}$ bins need to be defined. It is common and convenient to use an equidistant spacing (on the non-linear target scale) between a minimum and a maximum frequency f_{min} and f_{max} and usually $M^{(scale)} \leq M$. In most cases $f_{min}^{(lin)} = 0$ and $f_{max}^{(lin)} = 1/(2T_s)$ (converted to the target scale) is chosen to match the range of the linear scale spectrum. Some scales, such as the semitone scale or some versions of the Bark scale, however, require $f_{min}^{(lin)}$ to be greater than zero because $\Theta_{scale}(0)$ is not defined. With

$$f_{min}^{(scale)} = \Theta_{scale}(f_{min}^{(lin)}) \tag{2.50}$$

$$\text{and } f_{max}^{(scale)} = \Theta_{scale}(f_{max}^{(lin)}) \tag{2.51}$$

$$\text{and } f_{0}^{(scale)} = \frac{1}{M^{(scale)}}\left(f_{max}^{(scale)} - f_{min}^{(scale)}\right), \tag{2.52}$$

the frequencies $f^{(scale)} = F^{(scale)}(m^{(scale)})$ of the non-linear bins (equidistant on the target scale) are now given as:

$$F^{(scale)}\left(m^{(scale)}\right) = m^{(scale)} f_0^{(scale)} + f_{min}^{(scale)} \tag{2.53}$$

These frequencies $f^{(scale)}$ can be converted to a linear frequency scale with the inverse transformation function:

$$f^{(lin)}\left(m^{(scale)}\right) = \Theta_{scale}^{-1}\left(F^{(scale)}(m^{(scale)})\right) \tag{2.54}$$

Now the magnitudes $X_M^{(scale)}\left(m^{(scale)}\right)$ can be interpolated from the M linear scale magnitude bins $X_M(m)$ using various interpolation methods such as linear, cubic, or spline interpolation (Steffensen 2012).

Bark-Frequency scale The Bark frequency scale has been developed to numerically describe the loudness perception of human hearing. It was introduced by Zwicker (1961) and named after Heinrich Barkhausen, for his early achievements on subjective loudness measurements. The fundamental assumption in the definition is that a tone which is *perceived* as having twice the pitch than a reference tone also has twice the critical band rate (measured in Bark) of the reference tone. The original scale is defined from 0.2 Bark to 25 Bark. According to Zwicker (1961), below 500 Hz a tone with twice the frequency is perceived as having twice the pitch, while above 500 Hz a logarithmic rule applies for pitch perception versus frequency. This fact makes it hard to find a single exact analytic expression for the Bark frequency scale.

However, multiple analytical approximations of this scale exist. The most common one is the scale proposed by Traunmueller (1990). The critical band rate $f^{(bark)}$ (often called z in the literature) is computed via this core transformation:

$$z'(f) = \frac{26.81}{1 + \frac{1960}{f}} - 0.53, \tag{2.55}$$

followed by corrections for very low and very high frequencies:

$$f^{(bark)} = \Theta_{bark}(f) = \begin{cases} 0.85z'(f) + 0.3 & \text{if } z'(f) < 2 \\ z'(f) & \text{if } 2 \leq z'(f) \leq 20.1 \\ 1.22z'(f) - 0.22 \cdot 20.1 & \text{if } z'(f) > 20.1 \end{cases}. \tag{2.56}$$

The inverse transform is derived by inverting the above equations:

$$z' = \begin{cases} \frac{f^{(bark)}-0.3}{0.85} & \text{if} f^{(bark)} < 2 \\ f^{(bark)} & \text{if } 2 \leq f^{(bark)} \leq 20.1 \\ \frac{f^{(bark)}+0.22\cdot 20.1}{1.22} & \text{if} f^{(bark)} > 20.1 \end{cases}. \tag{2.57}$$

and

$$f = \frac{1960}{\frac{26.81}{z'+0.53} - 1}. \tag{2.58}$$

The Speex audio codec[10] uses a different version of the Bark scale given as:

$$\Theta_{bark,speex}(f) = 13.1 \arctan\left(0.74\frac{f}{1000}\right) + 2.24 \arctan\left(\frac{1.85}{10^8}f^2\right) + 10^{-4}f, \tag{2.59}$$

which is supposedly based on the original analytic approximation by Zwicker and Terhardt (1980):

$$\Theta_{bark,zwicker}(f) = 13 \arctan\left(0.76\frac{f}{1000}\right) + 3.5 \arctan\left(\left(\frac{f}{7500}\right)^2\right). \tag{2.60}$$

It is not trivial to invert the Speex approximation, and it is computationally more demanding to compute. For these reasons it is not considered any further in this thesis.[11]

Another approximation of the Bark frequency scale was suggested by Schroeder (1977):

$$\Theta_{bark,schroed}(f) = 6 \log\left(\frac{f}{600} + \sqrt{\left(\frac{f}{600}\right)^2 + 1}\right) \tag{2.61}$$

$$= 6 \sinh^{-1}\left(\frac{f}{600}\right), \tag{2.62}$$

[10] http://www.speex.org/.

[11] The SPEEX version of the Bark transformation is implemented in openSMILE as forward transformation only. Not all components will work, as most components require a backward scale transformation.

which has an inverse of:

$$f = \Theta^{-1}_{bark,schroed}\left(f^{(bark)})\right) = 600\,\sinh\left(\frac{f^{(bark)}}{6}\right). \tag{2.63}$$

Thereby the hyperbolic sine ($\sinh(x)$) and the inverse ($\sinh^{-1}(x)$) are defined as:

$$\sinh(x) = \frac{1}{2}\left(e^{x} - e^{-x}\right), \tag{2.64}$$

$$\sinh^{-1}(x) = \text{ arsinh }(x) = \log\left(x + \sqrt{x^2+1}\right). \tag{2.65}$$

Mel-Frequency scale Another analytical approximation of the critical band rate is the Mel-Frequency scale (Beranek 1949). According to Beranek (1949) and Young et al. (2006) it is defined as:

$$f^{(mel)} = \Theta_{mel}(f) = 1127 \cdot \log\left(1 + \frac{f}{700}\right). \tag{2.66}$$

In theory, one Bark corresponds to 100 Mel, although the scale approximations are different, and thus the scales cannot be converted directly one to the other.

The inverse transformation of the frequency $f^{(mel)}$ to a linear frequency f in Hz is given as:

$$f = \Theta^{-1}_{mel}(f^{(mel)}) = 700\left(e^{\frac{f^{(mel)}}{1127}} - 1\right). \tag{2.67}$$

Semitone-Frequency (Octave) scale For music analysis a frequency scale aligned with music notes is required for some descriptors. A semitone frequency scale for Western/European music is defined for 12 semitones per octave and the tones in the next octave have double the frequency of the tones in the previous octave. With this, the frequency f of each semitone $f^{(oct)}$ is given by:

$$f = f_{o0} \cdot 2^{\frac{f^{(oct)}}{12}}, \tag{2.68}$$

where f_{o0} is the frequency (in Hz) of the first note in the first octave, i.e., the frequency of $f^{(oct)} = 0$. Typically values of $f_{o0} = 27.5$ Hz or $f_{o0} = 55$ Hz are chosen, corresponding to an A_0 or A_1 note of modern pitch, where A_4 is standardised at 440 Hz (ISO16:1975 1975).

From Eq. (2.68) the forward frequency transformation from the linear frequency f in Hz to a real valued semitone number $f^{(oct)}$ can be derived:

$$f^{(oct)} = \Theta_{oct}(f) = 12\log_2\left(\frac{f}{f_{o0}}\right). \tag{2.69}$$

2.2.3.4 Band Spectra

Human hearing is redundant in a sense that masking takes place in both time and frequency (Zwicker and Fastl 1999). This means that of two sounds which have similar frequency the loudner one can (depending on the type of sound and levels) mask out the other one, rendering it inaudible. This suggests a reduction of the number of frequency bins M to a reduced number of bands B by combining bins within a defined band.[12] This is motivated by loudness perception.

This band division can be done linearly, i.e., each band having a constant width in Hertz, or non-linearly, where the bandwidth increases with the centre frequency of the band. Typically these non-linear bands are preferred because they are closer to non-linear frequency perception of the human hearing system. The methods for reducing M by combining bins into B bands, which will be described in the following, are general methods, which can work with any input frequency scale (linear or non-linear).

The general method for modelling masking and reducing the number of bands/bins at the same time is to define the power spectra of B band filters and then discretely convolve the M bin spectrum with each of the B band filters (cf. e.g., Hermansky 1990).[13] Assuming a general filter with a power spectrum $\Phi_b(m)$ which approximates the masking effects around the centre frequency of band b, the discrete convolution can be expressed as:

$$X_P(b) = \sum_{m=1}^{M} X_P(m)\Phi_b(m). \tag{2.70}$$

Next, a general function $g(x)$ defines the shape of the filter function on a general scale x. This scale can be any scale such as Bark scale, Mel scale, or linear frequency scale (Hertz (Hz)). In the ongoing, a linear Hz frequency scale is assumed, i.e., $x = f_{lin}$. Conversions to and from other scales can be implemented via Eqs. (2.48) and (2.49) from Sect. 2.2.3.3, respectively:

$$g^{(scale)}(f_{scale}) = g^{(lin)}(\Theta_{scale}^{-1}(f_{scale})), \tag{2.71}$$

and

$$g^{(lin)}(f_{lin}) = g^{(scale)}(\Theta_{scale}(f_{lin})). \tag{2.72}$$

The mapping of a continuous frequency value f_{scale} on scale *scale* to a real-valued bin index m_{scale} is given by Eq. (2.6) and the inverse by Eq. (2.7) (Sect. 2.1.2).

For linearly mapping a general filter shape $\Phi(m)$ defined for real-valued bin indices m': $g(m') = g(F^{-1}(f))$ to a discretised bin spectrum $\Phi(m)$ with integer bin

[12] For an implementation, see the `cMelspec` component in openSMILE and scale transformation functions in the smileUtil library.

[13] Band spectra can be computed in openSMILE with the `cMelspec` component, which—despite the name Melspec—can compute general band spectra for all supported frequency scales from a linear magnitude or power spectrum.

indices m, Eq. (2.73) can be applied:

$$\Phi(m) = \begin{cases} 0 & \text{for } m < m_l \\ \int\limits_{m-0.5}^{m+0.5} g(x)dx & \text{for } m \geq m_l \text{ and } m \leq m_u \\ 0 & \text{for } m > m_u \end{cases} . \tag{2.73}$$

Equation (2.73) assumes the bins to have a rectangular shape which is centred at the bin frequency $F(m)$.

In this thesis, rectangular and triangular filter shapes $g(f)$ are investigated, for which discrete versions are derived in the following. A rectangular filter function

$$g^{(rect)}(m') = \begin{cases} 1 & \text{for } m' \geq m'_l \text{ and } m \leq m'_u \\ 0 & \text{otherwise} \end{cases} , \tag{2.74}$$

with a lower cut-off frequency f_l and a corresponding real-valued bin number $m'_l = F^{-1}(f_l)$ as well as a upper cut-off frequency f_u with $m'_u = F^{-1}(f_l)$ is considered. Discrete lower and upper bound bin indices are obtained by rounding off the real-valued bin numbers: $m_l = \lfloor m'_l + 0.5 \rfloor$ and $m_u = \lfloor m_u l' + 0.5 \rfloor$. Now, Eq. (2.73) is applied and the rectangular filter can be expressed as:

$$\Phi^{(rect)}_{f_l, f_u}(m) = \begin{cases} 0 & \text{for } m < m_l \\ m_l - m'_l + 0.5 & \text{for } m = m_l \\ 1 & \text{for } m > m_l \text{ and } m < m_u \\ m'_u - m_u + 0.5 & \text{for } m = m_u \\ 0 & \text{for } m > m_u \end{cases} . \tag{2.75}$$

For the triangular filter, a real-valued centre (peak of the triangle) bin number $m'_c = F^{-1}(f_c)$ is required in addition to the real-valued bin number bounds m'_l and m'_u. A general triangle $g^{(tri)}(m')$ as function of the real valued bin number m' is then given as:

$$g^{(tri)}(m') = \begin{cases} 0 & \text{for } m' < m'_l \\ \dfrac{m' - m'_l}{m'_c - m'_l} & \text{for } m > m'_l \text{ and } m' \leq m'_c \\ \dfrac{m'_u - m'}{m'_u - m'_c} & \text{for } m > m'_c \text{ and } m' < m'_u \\ 0 & \text{for } m' > m'_u \end{cases} . \tag{2.76}$$

A triangular filter function $\Phi^{(tri)}_{f_l, f_c, f_u}(m)$ for integer bin indices is then given by integrating the function $g^{(tri)}(m')$ over each bin according to Eq. (2.73). Using $m_c = \lfloor m'_c + 0.5 \rfloor$, the discrete version of the triangular filter's power spectrum can be expressed as:

$$\Phi^{(tri)}_{f_l,f_c,f_h}(m) = \frac{1}{2}\begin{cases} 0 & \text{for } m < m_l \\ g^{(tri)}(m+0.5) & \text{for } m = m_l \\ g^{(tri)}(m+0.5) + g^{(tri)}(m-0.5) & \text{for } m > m_l \text{ and } m < m_c \\ \dfrac{g^{(tri)}(m_c-0.5) + g^{(tri)}(m'_c)}{1/\left(m'_c - m_c + 0.5\right)} & \\ \quad + \dfrac{g^{(tri)}(m_c+0.5) + g^{(tri)}(m'_c)}{1/\left(m_c + 0.5 - m'_c\right)} & \text{for } m = m_c \\ g^{(tri)}(m+0.5) + g^{(tri)}(m-0.5) & \text{for } m > m_c \text{ and } m < m_u \\ g^{(tri)}(m_u-0.5) & \text{for } m = m_u \\ 0 & \text{for } m > m_u \end{cases} \tag{2.77}$$

To account for the non-linear frequency resolution of the hearing system, filters with asymmetric slopes and a frequency dependent bandwidth proportional to the critical band rate are suggested by Zwicker and Fastl (1999). According to Zwicker (1970) such filters can be approximated by linear, triangular shaped filters on a non-linear frequency scale (e.g., Bark scale). The filters should be spaced equidistantly on the non-linear scale, which results in a non-linear spacing on the linear scale where the spacing of the filters increases with frequency. The bandwidth of the filters on the non-linear scale is constant. To apply such filters designed on a non-linear scale to a linear scale magnitude spectrum, the spectrum must first be scaled to a non-linear scale before the linear shape filters can be applied. This step has to be performed for every spectrum, i.e., for every audio frame. To save computation time, alternatively, the linear filters can be transformed from the non-linear (Bark or Mel, for example) scale to a linear Hertz scale:

The general filter shape $g^{(scale)}\left(m^{(scale)'}\right)$ on the non-linear frequency scale with $m^{(scale)'} = F^{-1}\left(f^{(scale)}\right) = F^{-1}\left(\Theta_{scale}\left(f^{(lin)}\right)\right)$ can be converted to the filter shape on a linear scale:

$$g^{(lin)}(m^{(lin)'}) = g^{(scale)}\left(F^{-1^{(scale)}}\left(\Theta_{scale}\left(F^{(lin)}\left(m^{(lin)'}\right)\right)\right)\right). \tag{2.78}$$

Equation (2.73) and derived versions thereof can now be used to estimate the shape of the filter on the linear frequency scale if $g^{(lin)}(m^{(lin)'})$ is substituted for $g(m')$ and the boundaries m_l, m_c, and m_u are replaced by the respective bin indices on a linear frequency scale:

$$m^{(lin)'} = F^{(lin)}\left(\Theta^{-1}_{scale}\left(F^{-1^{(scale)}}\left(m'_{scale}\right)\right)\right) \tag{2.79}$$

$$m^{(lin)} = \lfloor m^{(lin)'} + 0.5 \rfloor \tag{2.80}$$

It is important to note that the integer bin indices $m^{(lin)}$ for the bounds m_l, m_c, and m_u must be generated by converting the real-valued bin numbers $m^{(scale)'}_{l,c,u}$ (not the rounded bin indices $g^{(lin)}(m^{(lin)'})$) and rounding them according to Eq. (2.80).

The generalised filter equations derived here differ from the ones commonly implemented for Mel-frequency filter banks (Young et al. 2006) or Bark filter banks (Hermansky 1990), which are approximations optimised for improved computation time decades ago and have not been changed for compatibility reasons.

For compatibility with Young et al. (2006), the following triangular filterbank is used in this thesis for all non-linear frequency scales.[14] A filterbank of B filters with centres evenly spaced between f_{min} and f_{max} is created as follows: A constant bandwidth β for the filters on the target scale *scale* is assumed:

$$\beta = \frac{1}{B+1}\left(f_{max}^{(scale)} - f_{min}^{(scale)}\right). \tag{2.81}$$

The centre frequencies $f_c^{(scale)}(b)$ on the target scale for B filters $b = 1 \dots B$ and the lower and upper bounds ($b = 0$ and $b = B + 1$) are computed via:

$$f_c^{(scale)}(b) = f_{min}^{(scale)} + b\beta. \tag{2.82}$$

The power spectrum shape of each filter is then given by:

$$\Phi_b(m) = \begin{cases} 0 & \text{for } f^{(scale)}(m) \leq f_c^{(scale)}(b-1) \\ \dfrac{f^{(scale)}(m) - f_c^{(scale)}(b-1)}{f_c^{(scale)}(b) - f_c^{(scale)}(b-1)} & \text{for } f_c^{(scale)}(b-1) < f^{(scale)}(m) \leq f_c^{(scale)}(b) \\ \dfrac{f_c^{(scale)}(b+1) - f^{(scale)}(m)}{f_c^{(scale)}(b+1) - f_c^{(scale)}(b)} & \text{for } f_c^{(scale)}(b) < f^{(scale)}(m) \leq f_c^{(scale)}(b) \\ 0 & \text{for } f^{(scale)}(m) > f_c^{(scale)}(b+1) \end{cases}, \tag{2.83}$$

with

$$f^{(scale)}(m) = \Theta_{scale}\left(F^{(lin)}(m)\right). \tag{2.84}$$

Figure 2.2 shows plots of the power spectrum $\Phi(m)$ of two triangular filters with different centre frequencies which were designed as linear triangles on the Mel scale and converted to a discretised linear frequency scale (Eqs. (2.83) and (2.77)). A very low resolution for the DFT is assumed to highlight differences between the two implementations. 24 linear scale bins from $m = 1 \dots 24$ are shown in the plot, each bin 32.5 Hz wide. The x-axis labelling indicates the corresponding Mel frequency. It can be seen that both implementations produce nearly similar results, except for a small deviation at the centre frequency and at the left and right borders of the filter. It will be left to future work to empirically find out which implementation of the filters is better. Due to the minor differences, which are even less for a higher DFT resolution, it is fully justified that for the experiments in this thesis the simplified

[14] In openSMILE the `cMelspec` component implements these filterbanks for various frequency scales (not only Mel).

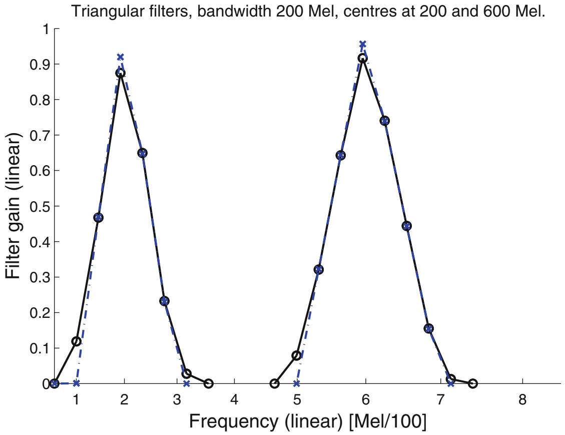

Fig. 2.2 Spectral shapes of two triangular filters designed with a centre frequency of 200 and 600 Mel and a symmetric (on the Mel scale) bandwidth of 200 Mel; *Dashed* (*blue*) *line* with (x) showing Eq. (2.83) and *black solid line* shows the version from Eq. (2.77) as derived in this thesis based on integration (Eq. (2.73))

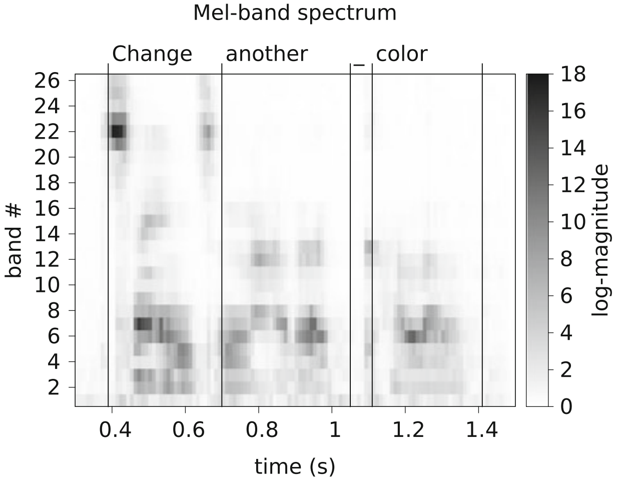

Fig. 2.3 Mel-band power spectrogram plot of a sample sentence from the AVIC database (Sect. 6.1.3); female speaker, words: "change another color"

implementation of Eq. (2.83) is used, which has also been used in other related work (e.g., Young et al. 2006). An example plot of a Mel-spectrogram obtained with this triangular filterbank is shown in Fig. 2.3.

2.2.3.5 Filterbank Spectra

An alternative to DFT based band spectra—as described in the previous sections—are time domain filterbank spectra. These are obtained by passing the unwindowed signal $x(n)$ through a bandpass filterbank with M filters and then computing the envelope of each of the filterbank outputs. To compute the envelope, windowing with size $N_f^{(m)}$ is performed on each of the filterbank outputs with a rate $T_f^{(m)}$ and the energy (Sect. 2.2.2) is computed for each window. The result is the spectral magnitude $X_M(m)$ for each band m at a rate of $T_f^{(m)}$. Thereby, for each band m a different envelope sampling rate $T_f^{(m)}$ as well as a different window size $N_f^{(m)}$ can be chosen. In practice, however, the sampling rate $T_f^{(m)}$ is a constant T_f for each band, with $T_f = 10\,\text{ms}$ or $T_f = 20\,\text{ms}$ in most cases.

Depending on the type, order, and number of bandpass filters, computation of such spectra can be slow when compared to FFT based spectra. Advantages, on the other side, are the improved frequency selectivity of time domain band-pass filters as well as the possibility of using a frequency dependent time resolution $T_f^{(m)}$.

For this thesis specifically Gabor filterbanks and gammatone filterbanks were implemented to approximate a critical band filterbank.[15] For practical implementation reasons, all filters were implemented as Finite Impulse Response (FIR) filters with discrete convolution. Thereby an impulse response $h(n)$ of finite length N_h is convolved with the input signal $x(n)$ to obtain the time domain output signal $y(n)$ (cf. Damelin and Miller 2011, p. 232):

$$y(n) = x(n) * h(n) = \sum_{m=-\frac{N_h}{2}}^{\frac{N_h}{2}} x(n-m) \cdot h(m). \tag{2.85}$$

To optimise the computational complexity, especially with long impulse response lengths N_h, the convolution can be carried out as a multiplication in the frequency domain. This approach is more efficient than the time domain implementation of the convolution because applying twice a FFT for long frames is more efficient than computing the full length convolution for every sample.

The one dimensional Gabor-filter impulse response $h_{gabor}(n)$ (for $N = -\frac{N}{2} \dots \frac{N}{2} - 1$) is given by (cf. Feichtinger and Strohmer 1998):

$$h_{gabor}(n) = e^{\beta\sqrt{2\pi}T_s n^2} \cos\left(2\pi f_c T_s n\right), \tag{2.86}$$

with bandwidth β (in Hz), centre frequency f_c (in Hz), and sampling period T_s.

The gammatone filter impulse response $h_{gammatone}(n)$ (for $N = -\frac{N}{2} \dots \frac{N}{2} - 1$) is given by (cf. Slaney 1993):

[15] In openSMILE the FIR filterbanks with Gabor, gammatone, high- and low-pass filters can be applied with the `cFirFilterbank` component.

$$h_{gammatone}(n) = aT_s n^{(o-1)} e^{-2\pi\beta n T_s} \cos(2\pi f_c n T_s), \tag{2.87}$$

where $n = 0 \ldots N_h - 1$,f_c is the centre frequency of the filter in Hz, β is the bandwidth of the filter in Hz, a is a linear gain factor which most often defaults to 1, and T_s is the sampling period.

The impulse responses for the gammatone and Gabor filters are infinite in time. In order to implement them with a FIR filter, a windowing must be performed, which introduces artefacts. This windowing is also referred to spectral shaping (Oppenheim et al. 1999). A windowing in the time domain corresponds to a convolution of the filter's frequency response with the spectrum of the windowing function. A good frequency response without side maxima can be achieved by multiplying the filter with a zero-endpoint Gaussian window $w_{gauss0}(n)$ (i.e., a Gaussian window scaled to have zero valued endpoints):

$$w_{gauss0}(n) = \frac{1}{1-\Gamma}\left(e^{-0.5\left(\frac{n}{\sigma\frac{N}{2}}\right)^2} - \Gamma\right) \tag{2.88}$$

with Γ being the minimum value of the unscaled Gaussian at the endpoints:

$$\Gamma = e^{-0.5\left(\frac{-\frac{N}{2}+1}{\sigma\frac{N}{2}}\right)^2} \tag{2.89}$$

An approximation of an ideal high or low-pass filter can be implemented via a windowed *sinc* function. The frequency response of the unwindowed, infinite duration *sinc* function is a perfect rectangle in the frequency domain, which corresponds to a low-pass filter. The steepness of the filter depends on the length of the window, i.e., the number of taps of the impulse response. The cut-off frequency f_c is determined by the *sinc* function, which leads to the following equation for the *sinc* impulse response h_{lowp} (for $N = -\frac{N}{2} \ldots \frac{N}{2} - 1$) of a low-pass filter:

$$h_{lowp}(n) = 2f_c \frac{sin\,(2\pi f_c T_s n)}{2\pi f_c T_s n}. \tag{2.90}$$

The high-pass filter can be constructed from the low-pass filter in the spectral domain by subtracting the low-pass filter spectrum from a flat spectrum which is the constant 1. In the time domain (for $N = -\frac{N}{2} \ldots \frac{N}{2} - 1$) this corresponds to:

$$h_{highp}(n) = \begin{cases} -h_{lowp}(n) & \text{for}\, n \neq 0 \\ -h_{lowp}(n) + 1 & \text{for}\, n = 0 \end{cases} \tag{2.91}$$

2.2.4 Spectral Descriptors

Now, a general spectrum $X(m)$ is defined, which can be a magnitude spectrum $X_M(m)$, a power spectrum $X_P(m)$, power spectral densities in dB, or a band or filterbank spectrum. With such a general spectrum $X(m)$ and a relation $f = F(m)$ (and $m = F^{-1}(f)$) between the linear frequency f in Hz and the bin index m (cf. Sect. 2.1.2), spectral statistics LLDs are defined in this section.[16] Most spectral descriptors can be computed from an arbirtrary sub-band range defined by the lower and upper bin indices m_l and m_u. The full range of the spectrum is covered when $m_l = 1$ and $m_u = M$. For the case of a constrained sub-band frequency range with respective lower and upper border frequencies of f_l and f_u, the respective integer valued bin indices are $m_l = \lfloor F^{-1}(f_l) + 0.5 \rfloor$ and $m_u = \lfloor F^{-1}(f_u) + 0.5 \rfloor$.

2.2.4.1 Band Energies

When computed from a high resolution FFT spectrum $X(m)$, this LLD allows to consider the energy in arbitrary bands. Similar to the concept of band spectra and triangular filterbanks, described in Sect. 2.2.3.4, the band energy is computed by a rectangular filter here, i.e., by summation of all the magnitudes within the interval $[f_l; f_u]$, where f_l and f_u are the lower and upper frequency bounds of the band, respectively. *Note:* This descriptor can be computed from any type of spectrum $X(m)$ in theory. However, the equations given below rely on a power spectrum $X_P(m)$ as input. Other spectral representations must be converted to a power spectrum, in order to be able to sum up the energies in each band with Eq. (2.92).

At the band borders interpolation is performed to consider partial bins appropriately, as derived in Sect. 2.2.3.4. With general lower and upper bound frequencies of f_l and f_u, respectively, according to Eq. (2.75) respective weighting factors $\alpha_l = m_l - m'_l + 0.5$ and $\alpha_u = m'_u - m_u + 0.5$ for the lower and upper integer bin indices m_l and m_u are found.

The energy for a band bounded by f_l and f_u is then computed as the following sum over the power spectrum X_P:

$$E_{f_l}^{f_u} = \alpha_l X_P(m_l) + \left(\sum_{m=m_l+1}^{m_u-1} X_P(m) \right) + \alpha_u X_P(m_u). \tag{2.92}$$

2.2.4.2 Spectral Slope

The overall shape of a spectrum $X(m)$ can be expressed by its linear slope. To compute the slope, a minimum quadratic error approximation of the spectrum $X(m)$ by a line

[16] In openSMILE these spectral descriptors can be extracted with the `cSpectral` component.

$$\hat{y} = ax + b \tag{2.93}$$

is attempted, where a is the slope, and b is the vertical displacement (Tamarit et al. 2008). In order to keep the following equations generic, the spectrum $X(m)$ is represented as a general function $y = f(x)$ with $y = X$ and $x = m$. The function $f(x)$ is defined for a finite set of N points x_i with $i = 0 \ldots N-1$.

The minimisation of the quadratic error e^2 between the linear approximation of the function and the function itself is expressed as:

$$\begin{aligned} e^2 &= \sum_{i=0}^{N-1} \left(y(x_i) - \hat{y}(x_i)\right)^2 = \sum_{i=0}^{N-1} (y_i - ax_i - b)^2 \\ &= \sum_{i=0}^{N-1} \left(y_i^2 - 2ax_iy_i - 2by_i + 2abx_i + a^2x_i^2 + b^2\right)^2 \overset{!}{=} min. \end{aligned} \tag{2.94}$$

From this, the following differential equations for a and b for the points x_i with $i = 0 \ldots N$ are obtained:

$$\frac{\delta}{\delta a} e^2 = \sum_{i=0}^{N-1} \left(-2x_iy_i + 2bx_i + 2ax_i^2\right) \overset{!}{=} 0, \tag{2.95}$$

$$\frac{\delta}{\delta b} e^2 = \sum_{i=0}^{N-1} (-2y_i + 2ax_i + 2b) \overset{!}{=} 0. \tag{2.96}$$

Rewritten as:

$$-\sum_{i=0}^{N-1} x_iy_i + b\sum_{i=0}^{N-1} x_i + a\sum_{i=0}^{N-1} x_i^2 \overset{!}{=} 0, \tag{2.97}$$

$$-\sum_{i=0}^{N-1} y_i + a\sum_{i=0}^{N-1} x_i + Nb \overset{!}{=} 0, \tag{2.98}$$

yields the solution for a:

$$a = \frac{N\sum_{i=0}^{N-1} x_iy_i - \sum_{i=0}^{N-1} x_i \sum_{i=0}^{N-1} y_i}{N\sum_{i=0}^{N} x_i^2 - \left(\sum_{i=0}^{N-1} x_i\right)^2} \tag{2.99}$$

For simplification, the following substitutions can be made in Eq. (2.99):

$$\begin{aligned} \sum_{i=0}^{N-1} x_i = \Sigma_x, \quad \sum_{i=0}^{N-1} y_i = \Sigma_y, \\ \sum_{i=0}^{N-1} x_i^2 = \Sigma_{x^2}, \quad \sum_{i=0}^{N-1} x_iy_i = \Sigma_{xy}, \end{aligned} \tag{2.100}$$

and Eq. (2.99) can be rewritten as:

$$a = \frac{N\Sigma_{xy} - \Sigma_x \Sigma_y}{N\Sigma_{x^2} - \Sigma_x^2} \tag{2.101}$$

For computational efficiency, when using a linear frequency scale, the following substitutions can be further made (by applying exponential sum properties from (Rade et al. 2000, p.189):

$$\Sigma_x = \frac{1}{2}N(N-1) \tag{2.102}$$

$$\Sigma_{x^2} = \frac{1}{6}N(N-1)(2N-1). \tag{2.103}$$

The spectral slope can be computed over the full range of the spectrum, or over a sub-band. When computing the spectral slope in a sub-band, the border points must be linearly interpolated from the neighbouring bins, if they do not exactly match a bin. Let us assume respective lower and upper band border frequencies of f_l and f_u, which correspond to real valued bin values $m'_l = F^{-1}(f_l)$ and $m'_u = F^{-1}(f_u)$. Then, the substitutions in Eq. (2.100) can be rephrased as (replacing $x_i = x_m = F(m)$ and $y_i = y_m = X(m)$):

$$\Sigma_x = f_l + \left(\sum_{m=\lceil m'_l \rceil}^{\lfloor m'_u \rfloor} F(m) \right) + f_u, \tag{2.104}$$

$$\begin{aligned} \Sigma_y = {} & X\left(\lfloor m'_l \rfloor\right) + \left(m'_l - \lfloor m'_l \rfloor\right)\left(X\left(\lceil m'_l \rceil\right) - X\left(\lfloor m'_l \rfloor\right)\right) \\ & + \left(\sum_{m=\lceil m'_l \rceil}^{\lfloor m'_u \rfloor} X(m) \right) \\ & + X\left(\lfloor m'_u \rfloor\right) + \left(m'_u - \lfloor m'_u \rfloor\right)\left(X\left(\lceil m'_u \rceil\right) - X\left(\lfloor m'_u \rfloor\right)\right), \end{aligned} \tag{2.105}$$

$$\Sigma_{x^2} = f_l^2 + \left(\sum_{m=\lceil m'_l \rceil}^{\lfloor m'_u \rfloor} F(m)^2 \right) + f_u^2, \tag{2.106}$$

$$\begin{aligned} \Sigma_{xy} = {} & f_l\left(X\left(\lfloor m'_l \rfloor\right) + \left(m'_l - \lfloor m'_l \rfloor\right)\left(X\left(\lceil m'_l \rceil\right) - X\left(\lfloor m'_l \rfloor\right)\right)\right) \\ & + \left(\sum_{m=\lceil m'_l \rceil}^{\lfloor m'_u \rfloor} F(m)X(m) \right) \\ & + f_u\left(X\left(\lfloor m'_u \rfloor\right) + \left(m'_u - \lfloor m'_u \rfloor\right)\left(X\left(\lceil m'_u \rceil\right) - X\left(\lfloor m'_u \rfloor\right)\right)\right) \end{aligned} \tag{2.107}$$

The spectral slope a computed with Eqs. (2.104)–(2.107) and Eq. (2.101) has the unit magnitude/Hz, if computed from a magnitude spectrum $X_M(m)$, energy/Hz if computed from a power spectrum $X_P(m)$, and dB/Hz if computed from log-power spectral densities $X_{P,norm}(m)$, for example.

Often, the spectral slope is computed from a linear frequency power spectrum over the full spectrum range (typically 0–8,000 Hz), as in, e.g., (Eyben et al. 2010a; Schuller et al. 2010, 2011, 2012a, 2013b). Alternatively, as suggested by Scherer et al. (2015), the spectral slope should be computed in three different bands, which are 0–1, 1–5, and 0–5 kHz. Thereby a logarithmic band spectrum is used, with constant linear bandwidth of 400 Hz.

2.2.4.3 Hammarberg Index

Besides computing the exact spectral slope directly, as was outlined above, features closely related to the spectral slope can be used. Tamarit et al. (2008) mention the *Hammarberg index* in this context. The measure was defined by Hammarberg et al. (1980) as the ratio of the strongest energy peak in the 0–2 kHz region to that of the strongest peak in the 2–5 kHz region. Hammarberg defined a fixed static pivot point of 2 kHz where the low and high frequency regions are separated. Symbolically the Hammarberg index η is defined here as:

$$\eta = \frac{\max_{m=1}^{m_{2k}} X(m)}{\max_{m=m_{2k}+1}^{M} X(m)}, \tag{2.108}$$

where m_{2k} is the highest spectral bin index where $f \leq 2$ kHz is still true.

According to more recent findings (Tamarit et al. 2008) it could be beneficial to pick the pivot point dynamically based upon the speaker's fundamental frequency. This is, however, on purpose not considered in this thesis because it would break the strictly static nature of all the feature extraction algorithms described. It will be left an open issue for future work where a multi-level feature extraction could be investigated, i.e., features are organized in different hirarchies and extraction algorithms for higher level features change their parameters according to decisions based on values of lower level features.

2.2.4.4 Alpha Ratio

Similar to the Hammarberg index, the Alpha Ratio (Patel et al. 2010) is defined as the ratio between the energy in the low frequency region and the high frequency region. More specifically, it is the ratio between the summed energy from 50 to 1000 Hz and 1 to 5 kHz, expressed as ρ_α:

$$\rho_\alpha = \frac{\sum_{m=1}^{m_{1k}} X(m)}{\sum_{m=m_{1k}+1}^{M} X(m)}, \tag{2.109}$$

where m_{1k} is the highest spectral bin index where $f \leq 1\,\text{kHz}$ is still true.

In applications of emotion recognition from speech, this parameter most often—like other spectral slope related paramters—is computed from a logarithmic representation of a band Long-Term Average Spectrum (LTAS) (cf. Scherer et al. 2015; Patel et al. 2010). However, the definition—in my opinion, holds for any sort of spectrum—and thus also for linear short time magnitude spectra, for example, as used in this thesis. Which spectral representation is best has to be determined empirically for every use-case.

2.2.4.5 Spectral Flatness

Spectral flatness ($S_{flatness}$) is computed as the ratio of the geometric mean of the spectral bins to the arithmetic mean of the spectral bins (Johnston 1988):

$$S_{flatness} = \frac{\sqrt[(m_u-m_l+1)]{\prod_{m=m_l}^{m_u} X(m)}}{\frac{1}{(m_u-m_l+1)} \sum_{m=m_l}^{m_u} X(m)}, \tag{2.110}$$

with lower and upper bound bin indices m_l and m_u, respectively. As suggested by the name this descriptor describes the flatness of a spectrum, i.e., it will have a higher value for a spectrum with strong peaks (e.g., a harmonic spectrum, but also a spectrum with non equidistant peaks such as multiple tones or modulated noise). This descriptor is part of the MPEG-7 audio content description standard (Manjunath et al. 2002).

2.2.4.6 Spectral Centroid

Spectral centroid ($S_{centroid}$) is computed as the centre of gravity of a spectrum $X(m)$ (cf. Peeters 2004):

$$S_{centroid} = \frac{\sum_{m=m_l}^{m_u} F(m)X(m)}{\sum_{m=m_l}^{m_u} X(m)}, \tag{2.111}$$

with lower and upper bound bin indices m_l and m_u, respectively. $F(m)$ is the frequency (in Hz) corresponding to bin m (cf. Sect. 2.1.2). It is correlated to the brightness or sharpness of an audio signal, according to Zwicker and Fastl (1999). Other studies (e.g., Kendall and Carterette 1996) suggest that the ratio between F_0 and the spectral centroid is better correlated to brightness than the spectral centroid alone. This descriptor is part of the MPEG-7 audio content description standard (Manjunath et al. 2002), also known as spectrum centroid.

The sums in Eq. (2.111) re-appear in equations for the spectral slope (Sect. 2.2.4.2), thus—in efficient code—spectral slope and spectral centroid should be computed in the same component in order to share the results and avoid computing these sums twice.[17]

2.2.4.7 Spectral Moments

Spectral moments include the second, third, and fourth order moments, which are named spectral variance (also referred to as spectral spread in the MPEG-7 standard (cf. Peeters 2004 and Manjunath et al. 2002)), the spectral skewness, and the spectral kurtosis, respectively. In order to compute these statistical moments, the spectrum $X(m)$ must be converted to a Mass Function (PMF) $p_X(m)$:

$$p_X(m) = \frac{X(m)}{\sum_{i=m_l}^{m_u} X(i)}. \tag{2.112}$$

Thereby usually the full spectral range is considered, i.e., $m_l = 1$ and $m_u = M$ and $X(m)$ is chosen to be the power spectrum, i.e., $X(m) = X_P(m)$.

Based on the above PMF, spectral variance $S_{variance}$ is defined as:

$$S_{variance} = S_\sigma^2 = \sum_{m=m_l}^{m_u} \left(F(m) - S_{centroid}\right)^2 p_X(m), \tag{2.113}$$

with the spectral centroid $S_{centroid}$ as defined in Sect. 2.2.4.6. The spectral standard deviation S_σ is given as:

$$S_\sigma = \sqrt{S_{variance}}. \tag{2.114}$$

Accordingly, spectral skewness $S_{skewness}$ is defined as:

$$S_{skewness} = \frac{1}{S_\sigma^3} \sum_{m=m_l}^{m_u} \left(F(m) - S_{centroid}\right)^3 p_X(m), \tag{2.115}$$

and spectral kurtosis $S_{kurtosis}$ as:

$$S_{kurtosis} = \frac{1}{S_\sigma^4} \sum_{m=m_l}^{m_u} \left(F(m) - S_{centroid}\right)^4 p_X(m). \tag{2.116}$$

[17] In openSMILE, this is implemented in the `cSpectral` component.

2.2.4.8 Spectral Entropy

The spectral entropy is similar to the spectral flatness, as it relates to the peakedness of the spectrum. Again, the spectrum must be converted to a PMF $p_X(m)$ using Eq. (2.112). Then, according to Misra et al. (2004), the spectral entropy $S_{entropy}$ is defined as:

$$S_{entropy} = -\sum_{m=m_l}^{m_u} p_X(m) \cdot \log_2 p_X(m). \tag{2.117}$$

The definition is based on the original definition of the Shannon Entropy by Shannon (1948).

2.2.4.9 Spectral Roll-Off Point

The $n\,\%$ spectral Roll-off Point (RoP) $S_{rop(n)}$ is defined as the frequency below which $n\,\%$ of the total energy is concentrated (generalised version of Peeters 2004). Thus, the spectral RoPs must be computed from the power spectrum $X_P(m)$. The total energy E (in the band from bin m_l to m_u) is computed as:

$$E = \sum_{m=m_l}^{m_u} X_P(m). \tag{2.118}$$

The $n\,\%$ RoPs is then estimated by iteratively increasing r in the following equation until the equation becomes valid:

$$\sum_{m=1}^{r} X_P(m) \geq \frac{n}{100} E. \tag{2.119}$$

The RoPs is then $S_{rop(n)} = F(r)$. Typical values for n are 95, 90, 75, and 50 %.

2.2.4.10 Psychoacoustic Sharpness

According to Zwicker and Fastl (1999), the spectral centroid is better correlated to the perceived sharpness of a sound, if it is computed on a Bark frequency scale and a frequency dependent auditory weighting for frequencies greater than 16 Bark (function $g(f^{(bark)})$) is performed on the magnitudes $X(m)$. This leads to the following definition of perceptual sharpness:

$$S_{sharpness} = 0.11 \frac{\sum_{m=1}^{M} \Theta_{bark}\big(F(m)\big) X(m) g\Big(\Theta_{bark}\big(F(m)\big)\Big)}{\sum_{m=1}^{M} X(m)}, \tag{2.120}$$

with

$$g\left(f^{(bark)}\right) = \begin{cases} 1 & \text{if} f^{(bark)} \leq 16 \\ \left(\frac{f^{(bark)}-16}{4}\right)^{\log 3/\log 2} + 1 & \text{otherwise} \end{cases}. \tag{2.121}$$

2.2.4.11 Spectral Differences

The spectral descriptors described so far are static descriptors in a sense that they can be applied to single spectra, e.g., short time spectra of a single frame. In order to consider changes in the spectrum over time, Spectral Difference (SD) features can be used. The general idea is to treat each short-time spectrum $X^{(k)}(m)$ at discrete time (frame index) k as a point X in an M-dimensional space and to compute the distance between two successive spectra. For distance computation the L_2 norm is considered in this thesis and the spectral difference $SD^{(k)}$ at frame index k is defined as:

$$SD^{(k)} = \sqrt{\sum_{m=m_l}^{m_u} \left(X^{(k)}(m) - X^{(k-1)}(m)\right)^2}. \tag{2.122}$$

This descriptor is useful for detecting sudden changes in the overall energy as well as sudden changes in the spectral shape of the signal without large changes in signal energy. It is widely used in Music Information Retrieval (MIR) for detection of instrument onsets (Masri 1996; Duxbury et al. 2002; Eyben et al. 2010b). A variation of the SD function is common in this field: the positive spectral difference $SD_+^{(k)}$, where only positive differences are considered in the sum:

$$SD_+^{(k)} = \sqrt{\sum_{m=m_l}^{m_u} \left(\frac{X^{(k)}(m) - X^{(k-1)}(m) + |X^{(k)}(m) - X^{(k-1)}(m)|}{2}\right)^2}. \tag{2.123}$$

This function emphasises areas of fast rising spectral energy, which in music corresponds to onsets of instruments and vocals. Decreasing energy is neglected, which is also of no importance of an onset detection algorithm, for example.

2.2.4.12 Spectral Flux

The spectral flux S_{flux} represents a quadratic, normalised version of the simple spectral difference (Sect. 2.2.4.11). With general normalisation coefficients μ_k the definition of spectral flux is as follows:

$$S_{flux}^{(k)} = \sum_{m=m_l}^{m_u} \left(\frac{X^{(k)}(m)}{\mu_k} - \frac{X^{(k-1)}(m)}{\mu_{k-1}}\right)^2. \tag{2.124}$$

The normalisation coefficients μ_k and μ_{k-1} can either be one[18] which corresponds to no normalisation, or chosen in a way that normalises the L_1 or L_2 norm of the spectrum vector to one:

$$\mu_k^{(L_1)} = \sum_{m=m_l}^{m_u} |X^{(k)}(m)|, \tag{2.125}$$

$$\mu_k^{(L_2)} = \sqrt{\sum_{m=m_l}^{m_u} \left(X^{(k)}(m)\right)^2}. \tag{2.126}$$

In the latter case (L_1 or L_2 vector normalisation) we talk about *normalised* spectral flux, otherwise, about *unnormalised* spectral flux.

2.2.4.13 Harmonicity

The *Harmonicity* descriptor implemented in openSMILE[19] describes the amount and quality of the harmonics in a signal, i.e., it can be used to discriminate harmonic and non-harmonic signals. It is thus related to the more common Harmonics-to-Noise ratio (see Sect. 2.2.13.3), however, not the same. Harmonicity is computed directly from a magnitude spectrum by applying a simple peak picking algorithm based on identification of local minima and maxima by looking at the 2 left and right neighbours of the current frame. Then, the ratio between the minima and the maxima in relation to the amplitude of the maxima is computed.

In detail, the maxima and minima of the magnitude spectrum $X_M(m)$ are found according to the following rule:

$$Max(m) = 1, \quad \text{if} \begin{cases} \quad\;\;\, X_M(m-2) < X_M(m) \\ \text{and } X_M(m-1) < X_M(m) \\ \text{and } X_M(m+1) < X_M(m) \\ \text{and } X_M(m+2) < X_M(m) \end{cases} \tag{2.127}$$

$$Min(m) = 1, \quad \text{if} \begin{cases} \quad\;\;\, X_M(m-2) > X_M(m) \\ \text{and } X_M(m-1) > X_M(m) \\ \text{and } X_M(m+1) > X_M(m) \\ \text{and } X_M(m+2) > X_M(m) \end{cases}. \tag{2.128}$$

[18] This is the current default in all openSMILE feature sets up to version 2.0. An option for normalisation might appear in later versions.

[19] In the `cSpectral` component.

Now, with an array $L(p)$ of the amplitudes of P alternating maxima and minima $p = 1 \dots P$ a sum Σ_P of maxima to minima distances is given as:

$$\Sigma_P = \sum_{p=2}^{P} |L(p) - L(p-1)|. \qquad (2.129)$$

The harmonicity descriptor HM is computed by normalising with the number of bins in the spectrum:

$$HM = \frac{\Sigma_P}{m_u - m_l + 1}, \qquad (2.130)$$

or by the total energy[20] of the frame in the sub-band:

$$\overline{HM} = \frac{\Sigma_P}{\sum_{m=m_l}^{m_u} X_M(m)}. \qquad (2.131)$$

2.2.5 Autocorrelation

Complementary to a short-time spectrum representation, the short-time Autocorrelation function (ACF) has a high resolution for low frequency periodicities within a frame or segment of interest. The autocorrelation describes the signal's self similarity at given discrete time lags $\tau \in [-\tau_{max} \dots \tau_{max}]$.

For a discrete signal $x(n)$ of infinite duration, the ACF can be computed in the time domain as:

$$ACF_e(\tau) = \sum_{n=-\infty}^{\infty} x(n)x(n+\tau). \qquad (2.132)$$

This energy based definition yields infinite values for signals of infinite duration. Thus, in practice the signal length is limited to N (frame length or period of an infinite duration signal):

$$ACF_e(\tau) = \sum_{n=0}^{N-1} x(n)x(n+\tau). \qquad (2.133)$$

Further, the energy ACF can be normalised by the length N in order to obtain a power ACF:

$$ACF_p(\tau) = \frac{1}{N} \sum_{n=0}^{N-1} x(n)x(n+\tau). \qquad (2.134)$$

[20] Enabled by the option *normBandEnergies* of the cSpectral component of openSMILE.

The definitions in Eqs. (2.133) and (2.134) assume that the signal is either periodic with N or has at least a duration of $N + 2\tau_{max}$. In the latter case (non-periodic), a signal of length N can also be zero-padded (i.e., filled with zeros) with $\tau_{max}/2$ zeroes at the left and $\tau_{max}/2$ zeroes at the right of the actual N samples.

Instead of computing the ACF via the sums in Eq. (2.133), the discrete FFT can be used to speed up computations (Wiener–Khintchine Theorem, cf. Wiener 1964 which bases on original publications in Wiener 1930, Khintchine 1934, and Levinson 1947a, b for discrete signals). Due to the fact that the autocorrelation can be seen as a convolution of $x(n)$ with a reversed version of $x(n)$, the autocorrelation of $x(n)$ can be expressed as multiplication in the frequency domain[21]:

$$ACF_e(x) = FFT^{-1}(FFT(x) \cdot FFT^*(x)), \tag{2.135}$$

with $FFT^*(x)$ denoting the complex conjugate of the discrete FFT of the signal $x(n)$ and $FFT^{-1}(.)$ representing the inverse discrete FFT. The above produces equivalent output to Eq. (2.133). This definition implicitly assumes the signal $x(n)$ to be continued periodically to the left and right. To avoid artefacts when the signal is not periodic in N, zero-padding to a length of $2N$ must be performed by placing $N/2$ zeroes to the left and $N/2$ zeroes to the right of $x(n)$. The valid range for τ is from $-N$ to $+N$ after zero-padding.

Designing the ACF in this way will avoid artefacts, however the magnitude of the ACF function will linearly decay towards the border because more zeroes will be part of the sum (Eq. (2.134)) for higher τ. To avoid this, two signals $x_1(n)$ and $x_2(n)$ can be defined, where $x_1(n) = x(n)$ for all $n \in [0; N[$ and $x_2(n) = x(n)$ for $n \in [\frac{N}{4}; \frac{3N}{4}[$. x_1 is zero-padded with $N/4$ zeroes to the left and right, and x_2 is zero-padded with $N/2$ zeroes to the left and right. The autocorrelation is then expressed as the cross-correlation between x_1 and x_2:

$$ACF_e(\tau) = \sum_{n=0}^{N-1} x_1(n)x_2(n+\tau). \tag{2.136}$$

This ensures a constant amplitude of the ACF for τ from $-N/4$ to $+N/4$ and a valid range of τ (i.e., non-zero output) from $-3N/4$ to $3N/4$.

In practice, most signals are not periodic with the frame length. Thus, symmetric zero-padding with $-N/2$ samples at both ends before applying the ACF is the most common method. The modification presented in Eq. (2.136) is not commonly used because it is not an autocorrelation in a strict sense any more and has slightly different properties. Instead, a τ-adaptive energy normalisation can be applied to obtain a corrected power ACF:

$$ACF_p(\tau) = \frac{1}{N-\tau} \sum_{n=0}^{N-1} x(n)x(n+\tau). \tag{2.137}$$

[21] ACF according to this equation is implemented in openSMILE in the cAcf component.

According to the above definitions, the ACF is defined for negative and positive values of τ. However, by definition the ACF is always symmetric and only the positive half, i.e., $\tau \in [0; \tau_{max}[$ is relevant.

The ACF constitutes a basis from which further descriptors can be computed. These include the Harmonics to Noise Ratio (HNR) (Sect. 2.2.13.3), fundamental frequency (Sect. 2.2.11), and ACF peak ratios (not considered here), for example.

2.2.6 Cepstrum

The Cepstrum (CEP) was introduced by Bogert et al. (1963). It is closely related to the ACF with respect to the way it is computed. Equation (2.135) is extended by a natural logarithm before the inverse FFT:

$$CEP_e = FFT^{-1}\{\ln\left(FFT(x) \cdot FFT^*(x)\right)\}, \tag{2.138}$$

$$CEP_e = FFT^{-1}\{\ln\left(|FFT(x)|^2\right)\}. \tag{2.139}$$

The effect of the logarithm is the temporal separation of the source and filter parts of the signal x in the Cepstrum for which the following will give a brief derivation and detailed explanation.

A source signal s is assumed which passes through a Linear Time Invariant (LTI) system which has an impulse response h. The signal y at the output of the LTI system is then defined by:

$$y(n) = \{s * h\}(n), \tag{2.140}$$

where $*$ denotes a convolution of the two discrete signals s and h, which is analytically defined as:

$$y(n) = \{s * h\}(n) = \sum_{n_h=0}^{N_h-1} s(n - n_h)h(n_h), \tag{2.141}$$

where N_h is the length of the impulse response h.

By exploiting properties of the FFT, the convolution in Eq. (2.140) can be expressed in the frequency domain as:

$$Y(m) = S(m)H(m), \tag{2.142}$$

where Y, S, and H are the results of the discrete FFT of the signals y, s, and h, respectively. If we now apply the power operator we obtain:

$$\begin{aligned} |Y(m)|^2 &= Y(m)Y^*(m) = S(m)S^*(m)H(m)H^*(m), \\ &= |S(m)H(m)|^2 = |S(m)|^2|H(m)|^2. \end{aligned} \tag{2.143}$$

Applying the natural logarithm to both sides of Eq. (2.143) yields:

$$\ln\left(|Y(m)|^2\right) = \ln\left(|S(m)|^2|H(m)|^2\right). \tag{2.144}$$

Now the algebraic property of logarithms that a multiplication of the arguments of the logarithm function is equivalent to a summation of the logarithms of the individual arguments can be exploited to transform the multiplicative (in the frequency domain) or convolutive (in the frequency domain) relation between the signals s and h into an additive relation:

$$\ln\left(|Y(m)|^2\right) = \ln\left(|S(m)|^2\right) + \ln\left(|H(m)|^2\right). \tag{2.145}$$

After inverse FFT the above becomes:

$$FFT^{-1}\{\ln\left(|Y(m)|^2\right)\} = FFT^{-1}\{\ln\left(|S(m)|^2\right)\} + FFT^{-1}\{\ln\left(|H(m)|^2\right)\}, \tag{2.146}$$

where

$$FFT^{-1}\{\ln\left(|Y(m)|^2\right)\} = FFT^{-1}\{\ln\left(|FFT(y)|^2\right)\} = CEP_e(y). \tag{2.147}$$

Thus, the Cepstrum of the output of a LTI system is the sum of the Cepstrum of the source signal and the Cepstrum of the system's impulse response.

In the case of a human speech signal the vocal tract can be seen—in a very simplified way—as a LTI system with the glottal excitation signal as the input $s(n)$ and the vocal tract together with the sound wave propagation from the mouth to adjacent air and the room as filter with the impulse response $h(n)$ (linear source-filter model) (Parsons 1987; Rosen and Howell 1991; Fant 1973). An example cepstrogram (plot) of a real speech signal is shown in Fig. 2.4. The impulse response h in this case (if room reverberation with high delay constants is not considered) is of rather short duration, thus, the Cepstrum is also short. In contrast, the excitation signal s is of longer duration, as it is either a periodic signal for voiced sounds (created by opening and closing of the vocal cords—in the ideal model it is a sequence of Dirac pulses) or a noise like signal for unvoiced sound (created by airflow through the glottis—white Gaussian noise in the ideal model). In the case of a periodic excitation this fact allows to separate the impulse response Cepstrum from the excitation function by taking the first K samples of the Cepstrum as impulse response and then filling them with zeroes leaving only the excitation signal. Thereby K must be smaller than the period length of the excitation signal. The fist two pulses of the excitation signal are clearly visible in Fig. 2.4.

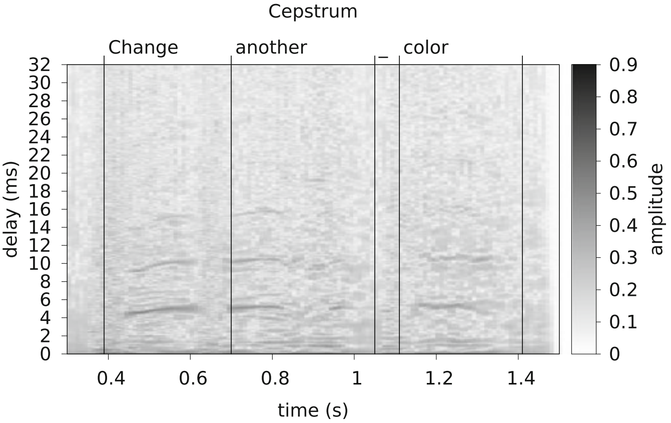

Fig. 2.4 Cepstrogram plot of a sample sentence from the AVIC database (Sect. 6.1.3); female speaker, words: "change another color"

2.2.7 Linear Prediction

Similar to Sect. 2.2.6, Linear Prediction (LP) analysis exploits the properties of the simple speech production model based on a LTI system. In this case the assumption is that if voiced speech is produced by a LTI system excited with periodic Dirac like signal, a sample at discrete time n will have a high correlation with the previous sample at $n-1$, i.e., one sample can be used to predict the following sample. Details on LP can be found in many other excellent sources, e.g., in a review by Makhoul (1975), in Ruske (1993), Furui (1996), and Schuller (2013). Therefore, in the following only the basic equations are summarised, which are required for implementation of LP for acoustic feature extraction.[22] In signal processing the term Linear Predictive Coding (LPC) is commonly used when talking about LP analysis.

In linear prediction an approximation $\hat{x}(n)$ of the original signal $x(n)$ is constructed by a weighted sum of p previous samples (Schuller 2013):

$$\hat{x}(n) = -\sum_{i=1}^{p} a_i x(n-i). \tag{2.148}$$

Thereby, p—the number of previous samples considered—is also referred to as the order of the linear predictor and the weighting coefficients a_i are called the linear predictor coefficients. The inversion in front of the sum is arbitrary, but simplifies the equations for the computation of the linear predictor coefficients.

Because the order p is finite and in practice much smaller than the length of the signal, an ideal estimation of an arbitrary signal $x(n)$ with Eq. (2.148) is not possible. An error term $e(n)$ is therefore included in Eq. (2.148) to capture the part of the signal

[22] In openSMILE linear predictive coding is supported via the cLpc component.

that cannot be modelled by the linear predictor (Schuller 2013):

$$x(n) = \hat{x}(n) + e(n) = -\sum_{i=1}^{p} a_i x(n-i) + e(n). \tag{2.149}$$

The signal $x(n)$ itself can now be included in the summation (starting at $i = 0$) which yields the following equation for the prediction error:

$$e(n) = \sum_{i=0}^{p} a_i x(n-i), \quad with \;\; a_0 = 1. \tag{2.150}$$

For acoustic parameter extraction based on LP analysis, primarily the predictor coefficients a_i are of interest. From Eq. (2.150) it can be seen that the predictor with the coefficients a_i can be seen as a linear transversal filter with the finite length (p) impulse response $h_{inv} = a_0, a_1, \ldots, a_p (a_0 = 1)$, which filters the input signal $x(n)$ and yields the output signal $e(n)$. Because this filter resembles the inverse of the vocal tract filter (which produces the speech signal $x(n)$ from the excitation signal $e(n)$), it is referred to as the *inverse* filter, hence the subscript *inv*. *Important:* It is to note here that the filter h_{inv} does not model only the vocal tract, it also includes the effects of the sound wave radiating from the mouth into the air, and the response of the room. For simplification, however, h_{inv} will be referred to as the vocal tract impulse response, and the reader is advised to keep in mind this approximation.

In the source-filter model of human speech production (cf. Sect. 2.2.6), the a_i coefficients correspond to the impulse response of the vocal tract and the prediction error signal $e(n)$ corresponds to the source excitation signal (Schuller 2013). The vocal tract impulse response depends on the configuration of the vocal tract, which in turn depends on the current phone that is being spoken. In particular, the impulse response contains vocal tract resonance frequencies (referred to as *formants*, cf. Deller et al. 1993 and Sect. 2.2.8).

In order to compute the predictor coefficients a_i, the constraint of quadratic minimisation of the error signal $e(n)$ is applied to Eq. (2.150). A sum squared error α is defined as:

$$\alpha = \sum_{n=0}^{N-1} e(n), \tag{2.151}$$

for the analysis interval $n = 0 \ldots N-1$, which is typically a short-time window (frame) of 10–20 ms. For a predictor of order p, α resembles the energy of the residual signal, which is related to the mean amplitude of the signal. It is thus also referred to as the LP gain.

From the quadratic minimisation condition:

$$\frac{\delta}{\delta x} \alpha \overset{!}{=} 0, \tag{2.152}$$

a system of linear equations is obtained, which can be solved for the p coefficients a_i with various methods (Ruske 1993).

2.2.7.1 LP Coefficients—Autocorrelation Method

The LP coefficients a_i are also referred to as the autoregressive (AR) coefficients because they are coefficients of an AR model. This section deals with the autocorrelation method for computing these AR coefficients (autoregressive modelling) as described and derived in more detail by Ruske (1993).

As a result of the derivation of the autocorrelation method, a recursive algorithm[23] for computing the predictor coefficients is found (according to Ruske 1993) as follows: The algorithm is initialised with a predictor of order $j = 0$, and with a sum quadratic prediction error of α_0:

$$\alpha_0 = r(0), \quad a_{00} = 1. \tag{2.153}$$

Then, a predictor of the next higher order j is iteratively estimated via the following equations, known as Levinson-Durbin recursion (cf. Levinson 1947b and Durbin 1960):

$$k_j = -\frac{1}{\alpha_{j-1}} \sum_{i=0}^{j-1} a_{i,j-1} r(j-i) \tag{2.154}$$

$$a_{jj} = k_j \tag{2.155}$$

$$a_{0j} = 1 \tag{2.156}$$

$$a_{ij} = a_{i,j-1} + k_j a_{j-i,j-1} \quad \text{with } i = 1 \ldots j-1 \tag{2.157}$$

$$\alpha_j = \alpha_{j-1}(1 - k_j^2), \tag{2.158}$$

where $j = 1 \ldots p$ indicates the iteration number, a_{ij} is the ith predictor coefficient after iteration j, $r(d)$ is the autocorrelation coefficient with associated lag d. The iteration terminates if a predictor of the requested order p is found, i.e., when $j = p$. The temporary variables k_j with $k \in [1; p]$ are—without further explanation at this point—named the reflection coefficients or the *partial correlation coefficients (PARCOR)* (cf. Ruske 1993).

The autocorrelation coefficients $r(d)$ for lags d are estimated via the autocorrelation function (cf. Sect. 2.2.5), assuming a window (frame) of N samples length:

$$r(d) = \sum_{n=d}^{N-1} x(n)x(n-d), \tag{2.159}$$

with $d \in [0; p]$.

[23] As implemented in openSMILE in the `cLpc` component.

2.2.7.2 LP Residual

The error signal $e(n)$ is also referred to as the LP residual. In general, it is computed from the speech signal by applying the inverse filter h_{inv} which is composed of the predictor coefficients (see Sect. 2.2.7). A preferred implementation of this filter is via a lattice filter structure. This is briefly outlined in this section.

The reflection coefficients k_j introduced in the previous section constitute an alternative representation of the LP filter and are restricted to the following condition (Ruske 1993):

$$|k_j| \geq 1. \tag{2.160}$$

Thus, they can be used to implement stable lattice filters for a forward vocal tract model filter (producing the speech signal from the excitation/error signal), and an inverse vocal tract model filter (producing the excitation/error signal from the speech signal).

For computing the residual $e(n)$ of $x(n)$, the inverse LP filter must be applied, of which the efficient lattice filter implementation is defined by the following equations (Ruske 1993):

$$\text{Initialisation:} \quad y_t^{(0)}(n) = y_b^{(0)}(n) = x(n) \tag{2.161}$$

$$\text{for } j = 1 \ldots p: \quad y_t^{(j)}(n) = y_t^{(j-1)}(n) + k_j \cdot y_b^{(j-1)}(n-1) \tag{2.162}$$

$$y_b^{(j)}(n) = y_b^{(j-1)}(n-1) + k_j \cdot y_t^{(j-1)}(n) \tag{2.163}$$

$$\mathit{Output}: \quad e(n) = y_t^{(p)}(n), \tag{2.164}$$

where p is the order of the linear predictor and k_j are the reflection coefficients (see Sect. 2.2.7.1).

2.2.7.3 LP Spectrum

The LP coefficients are related to the impulse response of the vocal tract (more exactly, the whole speech production system which includes, vocal tract, radiation into the air, and the room—however, for simplification it is referred to as vocal tract here), as mentioned before. The coefficients a_i constitute the impulse response of the inverse vocal tract filter (h_{inv}), which produces the excitation signal from the speech signal. In the source-filter model of speech production, however, the glottal excitation signal is convolved with the vocal tract system's impulse response h. Therefore, h must be obtained from h_{inv}. The spectrum H of h represents the frequency response of the vocal tract. In the following it is shown that H can be computed from the LP coefficients and is thus referred to as LP spectrum. Applying the DFT to h_{inv} yields (Schuller 2013):

$$H_{inv}(m) = DFT(h_{inv}). \tag{2.165}$$

The forward vocal tract filter H is defined by a recursive all-pole filter, which is obtained by inversion of H_{inv}:

$$H(m) = \frac{1}{H_{inv}(m)}. \tag{2.166}$$

Thus, the LP spectrum is defined as $H(m)$, which is, in words, the inverse of the DFT of the LP coefficients.

Given a predictor length p, a p-point DFT would be applied which yields $p/2$ frequency bins. For typical LP analysis p is rather small ($\approx$10–15), which leads to a bad frequency resolution in the LP spectrum. The resolution can be improved by non-linear interpolation (spline or *sinc*) or by zero-padding the LP coefficients to a length $N > p$, which is equivalent to the *sinc* interpolation (Oppenheim and Schafer 1975). The spectral resolution in the latter case is determined by N and the sampling rate of the original speech signal. For M bins $(2M - p)$ zeros need to be appended to the predictor coefficients.

2.2.7.4 Relation of LP Coefficients and the Cepstrum

A strong link between Cepstrum and LP AR coefficients exists due to their common relation to the impulse response of the vocal tract system in the linear speech production model. The link between cepstral coefficients $C(i)$ and the linear predictor coefficients a_i is defined by the following recursion (Young et al. 2006) (starting with $i = 0$):

$$C(i) = -a_i - \frac{1}{i}\sum_{j=1}^{i-1}(i-j)a_j C\,(i-j)\,. \tag{2.167}$$

2.2.7.5 Line Spectral Pairs

Besides the direct AR coefficient and the reflection coefficient representation of the LP coefficients, an alternate parametrisation as Line Spectral Pairs (LSPs) or Line Spectral Frequencies (LSFs) exists (Kabal and Ramachandran 1986). This representation is favoured for low bandwidth channel transmissions, as it was shown to be less sensitive to quantisation noise (Soong and Juang 1984; Kang and Fransen 1985).

According to Kabal and Ramachandran (1986), the LSPs are obtained by a decomposition of the LP coefficient polynomial in the z-domain ($H(z)$) with the help of the following two constructed polynomials[24]:

$$P(z) = H(z) + z^{-(p+1)}H(z^{-1}) \tag{2.168}$$

[24] In openSMILE the cLsp component implements LSP computation based on code from the Speex codec library (www.speex.org).

$$Q(z) = H(z) - z^{-(p+1)}H(z^{-1}), \tag{2.169}$$

$$\text{with } z = e^{j\omega} = e^{j2\pi f}, \tag{2.170}$$

where $P(z)$ and $Q(z)$ represent the vocal tract system with the glottis closed and opened, respectively (Schuller 2013). More details on the computation of LSPs are given by Kabal and Ramachandran (1986) and Schuller (2013) and are thus not repeated here. In short, the roots of the polynomials $P(z)$ and $Q(z)$ (of order $p + 1$) are determined empirically. The name Line Spectral Pairs stems from the fact that these roots are all complex symmetrical pairs with respect to positive and negative frequencies f (Schuller 2013; Furui 1996). In total, p roots can be found—the same number as LP coefficients.

The LSPs are related to the formants of the speech signal, which are estimated from the roots of the polynomial $H(z)$. Precisely, two LSFs enclose a formant as left and right boundaries.

2.2.8 Formants

Formants are resonance frequencies of the vocal tract system (Ruske 1993), which characterise phonemes, esp. vowels. They have also been discussed in the context of music, in particular the singing voice (Sundberg 1987). Hence, they are visible in speech spectra as maxima of the envelope. From this, one method for identification of formants is the application of a peak picking algorithm directly to speech power spectra. However, this approach suffers from distortions by the fundamental frequency and other peaks caused by other components of the speech production system (room transfer function, for example) as well as additive noise (Ruske 1993). The LP spectrum (Sect. 2.2.7.3) can be used to eliminate the influence of the fundamental frequency and to obtain a smoothed spectral envelope for formant estimation. An even more robust way of formant estimation bases on finding the roots of the LP coefficient polynomial (see Sect. 2.2.7), which will be outlined in the following section.[25] For all methods, again, one has to keep in mind that in the spectral envelope and also in the LP coefficients other resonance frequencies (room) might be contained and one must determine which peaks (or roots of the LP polynomial) belong to formants and which can be attributed to external influences (McCandless 1974).

Most recent methods build on the fundamental extraction principles outlined above, but consider temporal context for smoothing the formant trajectories and correcting local errors. For instance, the Viterbi algorithm is used to smooth trajectories (Yan et al. 2007) and/or Kalman filters are applied (Yan et al. 2007; Mehta et al. 2012). Few methods, such as (Glaser et al. 2010) implement alternative algorithms

[25] In openSMILE formant extraction is implemented via this method in the cFormant component, which processes the AR LP coefficients from the cLpc component.

for identifying formant trajectories directly from spectrograms, i.e., not basing on linear prediction. Temporal smoothing for formants is not considered in this thesis.

As for the nomenclature, the symbol F_i with $x \geq 1$ will be used for formants. This takes into account the fact that the fundamental frequency is denoted by the symbol F_0 and formants are, loosely speaking, higher orders of resonance frequencies excited by the fundamental frequency.

2.2.8.1 Estimation from Linear Prediction

When computing formant frequencies from LP coefficients, it is assumed that the formants are the resonance frequencies of the forward LP transfer function $H(m)$ (Eq. (2.165)). Thus, they represent zeros in the inverse transfer function, and hence can be determined from a factorisation of the inverse transfer function polynomial $H_{inv}(z)$ in the z-domain (McCandless 1974).

Given a set of predictor coefficients $h_{inv} = a_0, a_1, \ldots, a_p$ with $a_0 = 1$, the z-domain inverse transfer function of the LP filter can be expressed as (Schuller 2013; Ruske 1993):

$$H_{inv}(z) = 1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_p z^{-p}. \tag{2.171}$$

The factorisation of $H_{inv}(z)$ is given by:

$$H_{inv}(z) = \left(1 - \frac{p_1}{z}\right)\left(1 - \frac{p_1}{z}\right) \cdot \cdots \cdot \left(1 - \frac{p_M}{z}\right) \tag{2.172}$$

The factorisation is estimated with numerical methods, such as the Newton-Rhapson iteration (cf. Deuflhard 2011; Schuller 2013): the algorithm is initiated with an estimate of the first zero; then, the value of the polynomial and the numerical derivative at this location are computed. Iterative improvements of the estimate are computed by adjusting the location of the zero in the direction of the derivative, until a convergence criterion is reached (the delta from one iteration to the next is smaller than a threshold). A polynomial division is performed to remove this zero from the polynomial, and the algorithm is re-initialised for the next zero. This procedure is repeated until all zeros have been found, i.e., the degree of the remaining polynomial is 1. Because an error made for the first zero will propagate as an increased error to the subsequent zeros, it is recommended to refine the estimates of all the zeros in a small number of iterations on the whole polynomial.

The following algorithm to compute the formant frequencies from the transfer function poles is described loosely after the implementation in (Boersma 2001). Before computing the formant frequencies, the poles p_i are mapped into the unit circle by the following equation:

$$p_i = \frac{1}{p_i^*} \text{ if } |p_i| > 1. \tag{2.173}$$

Since all poles which are found for a polynomial $H_{inv}(z)$ with real-valued coefficients are symmetric on the imaginary axis, i.e., for each pole a complex conjugate partner pole exists, the poles with negative imaginary parts are discarded. Next, the phase of each pole p_i (with positive imaginary part) is converted to the frequency of the formant F_i:

$$F_i = \frac{1}{T_s 2\pi} \left| \arctan \left(\frac{Im(p_i)}{Re(p_i)} \right) \right| . \tag{2.174}$$

The corresponding bandwidth ("width of the spectral envelope peak") is estimated as:

$$F_i^{(bw)} = -\log \left(\frac{|p_i|}{T_s 2\pi} \right) . \tag{2.175}$$

2.2.8.2 Estimation from the Spectral Envelope

Formants can theoretically be estimated directly from the spectral envelope by peak picking. It is crucial though to choose an accurate method for estimating the spectral envelope. Basically two approaches exist:

1. Direct method: smoothing of the magnitude spectrum with a low-pass filter, or
2. computing the LP spectrum from LP AR coefficients.

The direct method has numerous disadvantages, hence, the LP spectrum method is the preferred way. The greatest disadvantage of the direct method is the influence of the fundamental frequency F_0 of voiced sounds. Because the range of possible F_0 values overlaps with the range of the first formant F_1, in some cases (especially for female voices) these two peaks might not be separable, or F_0 might be mistaken for F_1.

The LP spectrum (Sect. 2.2.7.3) contains a spectral envelope where the influence of F_0 has been completely removed by the linear predictor, if the length of the linear predictor $p \ll N_{F_0}^{(T)}$ ($N_{F_0}^{(T)}$ is the length of the fundamental frequency period in samples). Given that the LP spectrum has a sufficiently high resolution, the resonance frequencies of the vocal tract can be estimated by identifying peaks in the LP spectrum (McCandless 1974).

2.2.9 Perceptual Linear Prediction

The concept of Perceptual Linear Prediction (PLP) was introduced by Hermansky (1990). It extends the standard LP analysis by a psychoacoustic model of sound perception in the human auditory system. Hermansky (1990) defines the five steps of the PLP procedure:

1. Spectral analysis: 20 ms Hamming window with zero-padding to nearest larger power of 2 for FFT,
2. Critical-band power spectrum,
3. Equal-loudness pre-emphasis,
4. Intensity loudness power law,
5. Autoregressive modelling.

The result of the first four steps is a discrete auditory band spectrum $X_{p,aud}(b)$. With this, Hermansky (1990) suggests autoregressive modelling (step 5) with the autocorrelation method (as described in Sect. 2.2.7.1) to obtain AR coefficients. These coefficients can be then converted to a cepstral representation (Perceptual Linear Prediction Cepstral Coefficients (PLP-CC)) with Eq. (2.167). Similarly, PLP-CC could be computed directly from $X_A(m)$ by taking the natural logarithm of $X_A(m)$ and applying the Discrete Cosine Transformation (DCT). In this thesis, however, the original method via autoregressive modelling is adopted.[26]

The following sections describe the steps of PLP as they have been implemented in openSMILE for this thesis. Further, an extension to PLP, which accounts for temporal properties of speech is summarised in Sect. 2.2.9.5.

2.2.9.1 Critical-Band Spectrum

A critical-band power spectrum $X_P(b)$ is obtained for B Bark or Mel bands, as outlined in Sect. 2.2.3.4. Hermansky (1990) suggests the use of a Bark scale for frequency warping according to Eq. (2.62) (cf. Schroeder 1977).[27] Yet, any other non-linear frequency scale can also be used theoretically to obtain non-standard PLP-like coefficients.[28]

In this thesis triangular filters for the band spectrum are used, as implemented by Young et al. (2006). Although, Hermansky (1990) gives a piecewise linear shape for the band filters the triangular shape approximation has in practice been used more frequently due to the implementation provided by Young et al. (2006).

2.2.9.2 Equal-Loudness Pre-emphasis

The band spectrum $X_P(b)$ obtained in the previous step is now weighted by an equal loudness curve function $E(b)$ to attribute for the frequency dependent loudness sensitivity of the human ear:

[26] PLP via this method is implemented in openSMILE via the `cPlp` component.

[27] In openSMILE this Bark scale can be selected in the cMelspec component by setting the specScale option to 'bark_schroed'.

[28] openSMILE allows for this flexibility because the PLP procedure builds on a chain of components: cTransformFFT, cFFTmagphase, cMelspec (for the non-linear band spectrum), and cPlp (for equal loudness and intensity power law and autoregressive modelling and cepstral coefficients).

$$X_{P,eql}(b) = E(b) \cdot X_P(b) \tag{2.176}$$

The equal loudness function adopted by Hermansky (1990) is originally from Makhoul and Cosell (1976). It is given for a linear frequency f by:

$$E(f) = \frac{10^{31} \cdot \left((2\pi f)^2 + 56.8 \times 10^6\right)(2\pi f)^4}{\left((2\pi f)^2 + 6.3 \times 10^6\right)^2 \cdot \left((2\pi f)^2 + 0.38 \times 10^9\right)\left((2\pi f)^7 + 1.7 \times 10^{31}\right)}. \tag{2.177}$$

To evaluate the equal loudness function for frequencies $f^{(bark)}$ on the Bark scale, one substitutes f by:

$$f = \Theta_{bark}^{-1}\left(f^{(bark)}\right). \tag{2.178}$$

For the band power spectrum $X_P(b)$ the Bark scale band centre frequencies $f_c^{(bark)}(b)$ are used to compute the weights $E(b)$ for each band $b = 0 \ldots B - 1$. To speed up computations, the equal loudness weights can be applied directly to the triangular band filters (cf. previous section).

For compatibility with the Hidden Markov Toolkit (Young et al. 2006), the following approximation for the equal loudness curve is also investigated[29]:

$$E_{htk}(f) = \left(\frac{f^2}{f^2 + 1.6 \times 10^5}\right)^2 \frac{f^2 + 1.44 \times 10^6}{f^2 + 9.61 \times 10^6}. \tag{2.179}$$

The same conversions from linear to bark frequencies and to band indices as above can be applied to obtain the weighting factor for each band b.

2.2.9.3 Intensity Loudness Power Law (Compression)

To model humans' non-linear perception of sound power (Zwicker and Fastl 1999), a cubic root amplitude compression[30] is performed (Hermansky 1990), and a spectrum, referred to as *auditory spectrum* herein, is obtained:

$$X_{P,aud}(b) = X_{P,eql}(b)^{0.33}. \tag{2.180}$$

See Fig. 2.5 for an example auditory spectrogram. $X_{P,aud}(b)$ can be seen as a perceptually corrected band spectrum. The sum of the auditory spectrum over all bands is a perceptual *loudness* measure, which approximates the Zwicker loudness (Zwicker and Fastl 1999) and should be used for polyphonic wide-band signals as a substitute for the narrow-band loudness approximation (Sect. 2.2.2) in all cases.

[29] In openSMILE it is enabled by setting `htkcompatible` to 1 in the `cPlp` component.

[30] Configurable via the option `compression` in the openSMILE component `cPlp`.

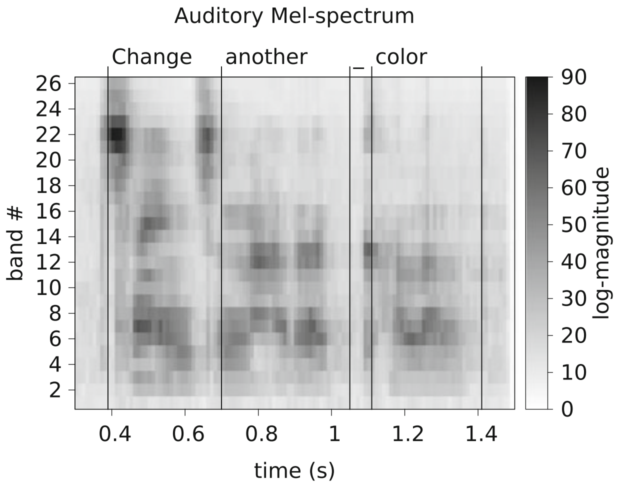

Fig. 2.5 Auditory spectrogram (based on 26-band Mel-band power spectrum) of a sample sentence from the AVIC database (Sect. 6.1.3); female speaker, words: "change another colour"

2.2.9.4 Autoregressive Modelling

The auditory spectrum $X_{P,aud}(b)$ is now used to estimate linear prediction coefficients, which—due to the perceptual/auditory nature of this spectrum—are called *perceptual* linear prediction (PLP) coefficients. These coefficients are estimated via the autocorrelation method, as described in Sect. 2.2.7.1. Instead of computing the autocorrelation coefficients $r(d)$ from the time domain signal $x(n)$, these coefficients are computed by applying the inverse DFT to the auditory spectrum. The auditory spectrum is a power spectrum, and thus, the inverse DFT of $X_{P,aud}(b)$ resembles an autocorrelation function. As only very few autocorrelation coefficients are required (order of the linear predictor 10), computing only the first few coefficients via a DFT is faster than using a FFT to compute the full autocorrelation.

2.2.9.5 Temporal Filtering: RASTA-PLP

RelAtive Spectral TrAnsform (RASTA) PLP is an extension to PLP which was first presented by Hermansky et al. (1992). It considers temporal properties of the human hearing and speech production systems. In particular, it exploits the fact that speech is primarily composed of modulations around 4 Hz (Zwicker and Fastl 1999; Hermansky et al. 1992). In RelAtive Spectral TrAnsform Perceptual Linear Prediction (RASTAPLP) a bandpass filter adapted to a range around 4 Hz is applied to the bands $X_P(b)$ before the auditory processing steps of PLP are applied.

An Infinite Impulse Response (IIR) filter for the RASTA bandpass is given by Hermansky et al. (1992) in the z-domain:

$$H(z) = 0.1z^4 \cdot \frac{2 + z^{-1} - z^{-3} - 2z^{-4}}{1 - 0.98z^{-1}}. \tag{2.181}$$

Adapting this filter for arbitrary lower (f_l) and upper (f_u) cut-off frequencies, and splitting it into an array of 5 FIR coefficients $\underline{a}$ and one IIR coefficient b gives the following set of equations for these filter coefficients:

$$\underline{a} = \left[\frac{2}{d}, \; \frac{-4o}{d}, \; 0, \; -\frac{-4o}{d}, \; -\frac{2}{d}\right], \tag{2.182}$$

$$b = 1 - \sin\left(2\pi f_l T_f\right), \tag{2.183}$$

where the helper variables o and d are defined as:

$$o = \cos\left(2\pi f_u T_f\right), \tag{2.184}$$

$$d = \sqrt{10\left(32o^2 + 8\right)}. \tag{2.185}$$

T_f is the period (in seconds) of the frames $X_P(b)$ (typically 10 ms).

Figure 2.6 shows the effect of the RASTA filter applied to the auditory spectrum shown in Fig. 2.5.

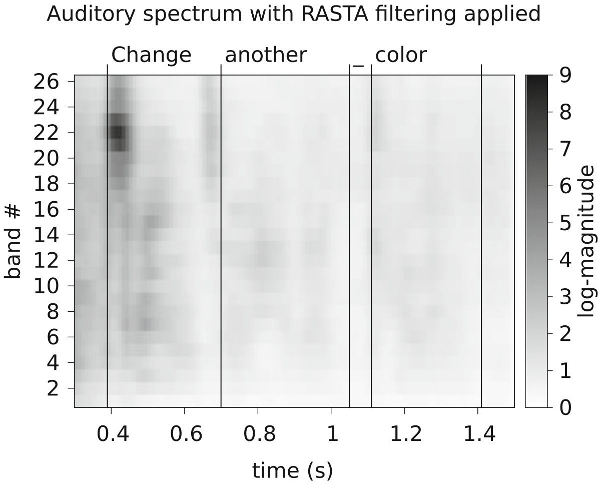

Fig. 2.6 RASTA filtered auditory spectrogram (based on 26-band Mel-band power spectrum) of a sample sentence from the AVIC database (Sect. 6.1.3); female speaker, words: "change another colour"

2.2.10 *Cepstral Features*

The idea of separating source and filter parts of a speech signal by Cepstral analysis, i.e., by applying a logarithm to the spectral magnitudes (cf. Sect. 2.2.6) is exploited by the group of cepstral features. These features are among the most successful and probably the most often used acoustic features for Automatic Speech Recognition (ASR) (Young et al. 2006; Rabiner and Juang 1993). They have also been applied successfully to various other audio tasks, such as speaker identification (Martinez et al. 2012), acoustic gunshot detection (Suman et al. 2014), music mood recognition (Nalini and Palanivel 2013), acoustic geo-sensing (Schuller et al. 2013a), and Computational Paralinguistics (Schuller 2013).

The general procedure for computing cepstral features is basically the same procedure as for computing the Cepstrum (Sect. 2.2.6):

1. Power spectrum representation (typically non-linear band spectrum)
2. Application of logarithm to power spectrum
3. Inverse spectral transformation (typically DCT)
4. Filtering of cepstral coefficients, called *liftering*.

However, in contrast to Sect. 2.2.6 for the cepstral descriptors discussed below, a non-linear frequency scale band spectrum serves as basis for Cepstrum computation. Further, the inverse FFT is replaced by a DCT.

In the following two sections the most prominent cepstral descriptors, namely MFCC and PLP-CC are described.

2.2.10.1 Mel-Frequency Cepstral Coefficients

With the building blocks described in the previous sections, MFCC as used by Young et al. (2006) are quickly described[31]: A 26 band spectrum $X_P^{(mel)}(b)$ (Sect. 2.2.3.4) is computed from a linear scale magnitude or power spectrum using triangular filters which are equidistant on the Mel-frequency scale and have 50 % overlap, i.e., the left and right endpoints of each triangular filter match the centres of the left and right bands. The filterbank is typically designed for the range from 20 Hz (in order to avoid the influence of DC components) to 8,000 Hz, in which case the bands have centre frequencies and half bandwidths as listed in Table A.1 in Appendix.

Next, the logarithm is applied to $X_P^{(mel)}(b)$:

$$X_P^{(log,mel)}(b) = \begin{cases} \log\left(X_P^{(mel)}(b)\right) & \text{if } X_P^{(mel)}(b) < X_{floor} \\ X_{floor} & \text{otherwise} \end{cases}. \tag{2.186}$$

[31] In openSMILE MFCC are computed via `cMelspec` (taking FFT magnitude spectrum from cFFTmagphase as input) and `cMfcc`.

The value X_{floor} is used to floor very small values (resulting from quantisation noise) in order to avoid large negative log values. If the input samples have been scaled to a range $[-1; +1]$, then $X_{floor} = 10^{-8}$ is assumed as default.[32]

On the log Mel-spectrum $X_P^{(log,mel)}(b)$, a Discrete Cosine Transformation Type-II (DCT-II) is performed (Young et al. 2006),[33] resulting in K Mel-cepstral coefficients $C^{(mel)}(k)$:

$$C^{(mel)}(k) = \sqrt{\frac{2}{B}} \sum_{b=0}^{B-1} X_P^{(log,mel)}(b) \cos\left(\frac{\pi k}{B}\left(b + \frac{1}{2}\right)\right). \tag{2.187}$$

Typically $K = 12 \ldots 16$ is chosen for most speech and music tasks. The advantage of these Cepstral coefficients is that they are decorrelated due to the orthogonal base of the DCT transformation. This has made them very popular for the use in Hidden Markov Model (HMM) systems with diagonal covariance matrices (Young et al. 2006).

In the final step, the Cepstral coefficients $C^{(mel)}(k)$ are filtered in a way to emphasize lower order coefficients. Because this happens in the Cepstral domain, it is called—in analogy to the wordplay of Cepstrum versus spectrum—*liftering*. Given a liftering coefficient L, the liftering is expressed as a weighting with a biased sine function:

$$C^{(mel)'}(k) = C^{(mel)}(k)\left(1 + \frac{L}{2}\sin\frac{\pi k}{L}\right). \tag{2.188}$$

The final coefficients $C^{(mel)'}(k)$ are those that are widely known and used as MFCC.

2.2.10.2 PLP Cepstral Coefficients

Similar to the MFCC, Cepstral coefficients can be computed based on the auditory spectrum of PLP analysis (Sect. 2.2.9) or RASTA-PLP (Sect. 2.2.9.5). These can either be computed in analogy to the MFCC by using the auditory spectrum from the PLP procedure instead of the log Mel-spectrum, or by applying the cepstral recursion from Eq. (2.167) to the PLP AR coefficients from Sect. 2.2.9.4. The latter method is the one implemented by Young et al. (2006) and in openSMILE.[34]

[32] In openSMILE the floor value is also 10^{-8} by default, and 1 when `htkcompatible=1` in `cMfcc`.

[33] Please note, that the DCT equation given in Young et al. (2006) and here differ because Young et al. (2006) start the summation at $b = 1$ for the first Mel-spectrum band, while here the first band is set at $b = 0$.

[34] PLP-CC can be computed in openSMILE by creating a chain of `cFFTmagphase`, `cMelspec`, and `cPlp` and setting the appropriate options for cepstral coefficients in the `cPlp` component.

2.2.11 Pitch

The previous descriptors have focussed on the vocal tract transfer function and have ignored the excitation signal. However, for applications besides ASR—such as Computational Paralinguistics and Music Information Retrieval—prosody (i.e., tonality, melody, dynamics of speech) plays a major role (Batliner et al. 2007; Schuller 2013). Prosody is mainly expressed through pitch and loudness.

This section deals with the estimation of pitch, or for most methods, an approximation of pitch via estimation of the fundamental frequency F_0. Pitch is a perceptual term which refers to the perceived tonality, i.e., frequency of a tone (Zwicker and Fastl 1999). It is related to the F_0, i.e., the lowest frequency in the harmonic series of the tone, however, not identical. Various effects contribute to the perception of pitch, leading to phenomena where the pitch of a tone is potentially perceived as higher or lower than the actual fundamental frequency. The most prominent case is the case of the missing fundamental (Zwicker and Fastl 1999). If a tone of 100 Hz (including higher order harmonics) is transmitted over a channel with a high-pass or band-pass characteristic (e.g., a telephone line), then the first harmonic might be the dominant fundamental frequency on the receivers side, while the actual fundamental of 100 Hz has been almost fully removed from the signal by the channel. Our ear reconstructs the original pitch based on the structure of the harmonics, which are nonetheless 100 Hz apart and not 200 Hz as they would have to be for a tone which actually has a pitch of 200 Hz.

Many automatic extraction methods aim at detecting periodicities in the signal, and thus constitute estimators of the fundamental frequency and not necessarily of pitch. Despite this fact, the topic is often commonly referred to as pitch detection in the literature (Hess 1983). Thus, the methods are also referred to as Pitch Detection Algorithms (PDAs) in the literature and in this thesis. In general two groups of PDAs exist: methods operating in the time domain, and methods operating in the short-time domain (frames) (cf. Schuller 2013). Short-time domain methods provide an F_0 estimate for each frame (typically 20–60 ms), which represents an average over ≈2–10 individual F_0 periods. Time domain methods detect individual fundamental periods directly from the waveform, and thus have a better temporal resolution than short-time domain methods, which is required, e.g., for estimation of micro-perturbations of F_0 (Jitter, cf. Sect. 2.2.13.1 and Schuller 2013). However, the frame-wise estimate from a short-time domain method can be used to initialise a refined search in the time domain, as will be shown in Sect. 2.2.13. Most PDAs also provide a measure of voicedness, the so called voicing probability p_v, which indicates how close the signal is to an ideal harmonic signal (high probability) or to a noise like signal (low probability). In the case of the time domain methods (Sect. 2.2.11.1), an additional method for estimating the voicing probability must be used, e.g., via autocorrelation (Sect. 2.2.11.2). Since the value of F_0 is not well defined for unvoiced parts of the signal, a value of 0 will be returned by all PDAs implemented in the scope of this thesis if p_v of a frame is smaller than a pre-defined threshold (specific for the respective PDA used).[35]

In the ongoing, four methods for pitch detection are described: in the time domain via peak picking (Sect. 2.2.11.1), via short-time autocorrelation (Sect. 2.2.11.2), and short-time Cepstral analysis (Sect. 2.2.11.2), as well as a spectral method based on subharmonic summation (Sect. 2.2.11.4). All four methods provide estimates for F_0 on a frame or sub-frame level, without considering the context of neighbouring frames. The context of a frame can be used, however, to eliminate errors a PDA makes in the presence of noise-bursts or irregular phonation. Such a context-based smoothing using the Viterbi algorithm is presented in Sect. 2.2.11.5. More advanced pitch detection algorithms such as Talkin (1995) and Cheveigne and Kawahara (2002) are not considered in this thesis, as they are not well suited for simple, fast, and efficient real-time on-line use due to their more complex smoothing methods and context requirements.

2.2.11.1 Time-Domain Based Estimation

The earliest methods for pitch detection were based on algorithms which attempted to directly estimate the fundamental period T_0 of the speech signal from the time domain waveform by smoothing and peak picking as reviewed by e.g., Ruske (1993) and Schuller (2013).

Since the fundamental period can only be determined for voiced signals, for each frame a decision is made, whether the frame contains a voiced (harmonic) or unvoiced (noise-like) signal, based on, e.g., the ZCR (Sect. 2.2.1.1) or any other suitable method.

In voiced speech segments, maxima which have a certain minimum and maximum distance are found: for a range of expected F_0 values from $F_{0,\,min}$ to $F_{0,\,max}$ a search window of length

$$T_{0,\,max} = \frac{1}{F_{0,\,min}} \tag{2.189}$$

and

$$T_{0,\,min} = \frac{1}{F_{0,\,max}}, \tag{2.190}$$

is assumed and the position of the maximum positive value within this window is found. If multiple values which are of approximately equal height are found, the first one is used. The position of this value is stored, and the start of the search window is shifted to $T_{0,\,min}$ samples after this position and the search is carried out for the next peak until the end of the voiced segment is reached.

[35] In openSMILE this behaviour is implemented in the pitch smoother components and in the `cPitchACF` component; the output F_0 final contains F_0 with values forced to 0 for unvoiced regions. See the documentation for more details.

The accuracy of such a method is very sensitive to all sorts of signal distortions, especially additive noise, digital clipping, and high-pass transfer characteristic channels. The noise robustness can be improved by low-pass filtering the signal with a cut-off frequency of $F_{0,\,max}$ or by applying the method to the (low-pass filtered) LP residual signal. The method fails, however, to detect the correct F_0 when the actual F_0 is missing (virtual pitch), or other harmonics are more dominant than F_0 (influence of strong first and second formants). The advantage, on the other hand, is that locations of individual pitch periods are obtainable (Schuller 2013).

2.2.11.2 Autocorrelation Based Estimation

The ACF of a frame (cf. Sect. 2.2.5) can be used to robustly estimate the fundamental frequency (Rabiner 1977; Boersma 1993). Peaks are found in the ACF at the period lengths of the harmonics of a harmonic signal. The ACF method assumes that T_0 is given by the location highest peak in the ACF in the search window from $T_{0,\,min}$ to $T_{0,\,max}$ (from the previous section). The amplitude of this peak, normalised by the amplitude of the 0th ACF coefficient gives a measure for the voicedness of the signal, i.e., the similarity of the pitch periods in the given frame. Thus, the probability of voicing p_v is estimated from the ACF as follows:

$$p_v = \frac{ACF_{max}}{ACF_0}. \tag{2.191}$$

where ACF_{max} is the maximum value in the range $T_{0,\,min} \ldots T_{0,\,max}$ and ACF_0 is the frame energy (0th coefficient of the ACF).

2.2.11.3 Cepstrum Based Estimation

In early studies related to this thesis (e.g., Schuller et al. 2009a, 2010), a PDA based on both autocorrelation and Cepstrum was used. Thereby the frequency F_0 is determined by the Cepstrum based method described in the following, and the probability of voicing is determined by the ACF method described in the previous section.

In a Cepstrum, as described in Sect. 2.2.6, the excitation signal (source) and the impulse response related to the filter transfer function of the linear model of speech production are overlaid in an additive way. Because the impulse response rapidly decays (due to the size of the human vocal tract), the higher order cepstral coefficients contain approximatively only the source signal. In the case of voiced speech signals, this is a Dirac impulse series. Due to the limited size of the analysis frame, often only one (for male voices) or two (for female voices) Dirac impulses are contained in the Cepstrum. F_0 is thus again determined by finding the location of the highest peak in the range $T_{0,\,min}$ to $T_{0,\,max}$ (the Quefrency axis of the Cepstrum has the units of seconds, thus the location of the peak corresponds directly to the fundamental period T_0 in seconds) (Noll 1967).

For this thesis, a hybrid autocorrelation/Cepstrum pitch detector has been implemented in openSMILE (Eyben et al. 2010a) and openEAR (Eyben 2009a).[36] First the probability of voicing is estimated from the ACF as described by Eq. (2.191). Next, F_0 is computed from the Cepstrum by searching for prominent peaks (extreme values) of the Cepstrum in the range $T_{0,\,min} \ldots N$ and favouring peaks with a lower T. To this extent, the mean value of the Cepstrum over the full range is computed (C_{mean}) and the absolute maximum value C_{max} in the range $T_{0,\,min} \ldots N$ is found. Then, a peak search starting at $T = T_{0,\,min}$ up to $T = N$ is performed, and the first peak found which meets the following criterion is chosen as T_0 peak:

$$C(T_0) > 0.6\,(C_{mean} + C_{max})\,. \tag{2.192}$$

The voicing probability p_v is estimated from the ACF via Eq. (2.191). In Fig. 2.8 (left), the pitch computed with this algorithm from the Cepstrum in Fig. 2.4 is shown. Octave jumps (double the actual frequency), discontinuities, and unvoiced segments which are wrongly classified as voiced (before 'change' and at the end and after 'color') are visible in the plot.

2.2.11.4 Subharmonic Summation (SHS)

A pitch detection method based on human perception principles (cf. Terhardt's virtual pitch theory (Terhardt 1974, 1979; Zwicker and Fastl 1999)) is shown in (Hermes 1988).[37] The method makes use of the harmonic structure of a signal to identify the correct pitch, even if the fundamental frequency is missing. It is thus called Subharmonic Summation (SHS) by the author. This method presented by Hermes (1988) is numerically more efficient compared to other earlier methods, such as the spectral-compression method (Schroeder 1968; Noll 1970) and the spectral-comb method (Martin 1982), which both also exploit the subharmonic structure. Therefore, the method by Hermes (1988) is adapted in this thesis and implemented in openSMILE. The following text briefly summarises the method.

First, the audio signal $x(n)$ is windowed by applying a 60 ms Gaussian window with $\sigma = 0.4$. The Gaussian window is chosen because of its Gaussian spectral shape with no side maxima, i.e., it does not distort the subharmonic structure. The larger size (60 vs. 40 ms as by Hermes 1988) is chosen to compensate for the flatter slope of the Gaussian window (as compared to the Hamming window), and in order to not distort low male pitch frequencies down to 50 Hz by the windowing.

[36] In the `cPitchACF` component, which requires combined ACF and Cepstrum input from two instances of the `cAcf` component.

[37] The method is implemented in openSMILE in two components: `cSpecScale` which performs spectral peak enhancement, smoothing, octave scale interpolation, and auditory weighting; `cPitchShs` which expects the spectrum produced by `cSpecScale` and performs the shifting, compression, and summation as well as pitch candidate estimation by peak picking.

A magnitude spectrum $X_M(m)$ is computed for the window. From there, the steps of the SHS algorithm (Hermes 1988) are as follows:

(1) Spectral peak enhancement (cf. Martin 1981)
(2) Spectral smoothing (optional)
(3) Interpolation from linear- to log-scale spectrum
(4) Auditory weighting
(5) Summation of shifted versions of the spectrum
(6) Peak picking

For (1), all local maxima are detected and their positions are stored. Then, all spectral bins which are more than two frames away from a local maximum are set to 0, to preserve only the peaks and their immediate surroundings. The spectrum $X_M^{(peak)}(m)$ is then smoothed (2) by convolving with a symmetric 3-tap filter in the spectral domain:

$$X_M^{(smooth)}(m) = \frac{1}{4}X_M^{(peak)}(m-1) + \frac{1}{2}X_M^{(peak)}(m) + \frac{1}{4}X_M^{(peak)}(m+1). \tag{2.193}$$

Next, the spectrum is transformed to a logarithmic octave (log_2) frequency scale (cf. Sect. 2.2.3.3). $B = M$ points on the target scale are assumed here, in contrast to (Hermes 1988), where a fixed number of $B_{oct} = 48$ points per octave is used. Spline interpolation (cf. Dalquist et al. 1974) is used to compute the magnitudes at the frequencies belonging to the log-scale bins $X_M^{(oct)}(m^{(oct)})$ (3). The minimum and maximum frequency (f_{min} and f_{max}) of the target scale are set to 20 Hz and the maximum frequency of the source scale, respectively. The number of octaves N_{oct} is then given as:

$$N_{oct} = \log_2\left(\frac{f_{max}}{f_{min}}\right) \tag{2.194}$$

and the number of points per octave computed as:

$$B_{oct} = \frac{B}{N_{oct}}. \tag{2.195}$$

In order to model the sensitivity of the human auditory system for low frequencies, a raised arctangent function is applied to the log-scale spectra $X_M^{(oct)}\left(m^{(oct)}\right)$ as weighting function $W\left(m^{(oct)}\right)$ (with $m^{(oct)} = 0 \ldots B-1$) (4):

$$W(m^{(oct)}) = 0.5 + \frac{1}{\pi}\arctan\left(3\frac{m^{(oct)}+1-\alpha_s}{B_{oct}}\right), \tag{2.196}$$

with

$$\alpha_s = B_{oct}\log_2\left(\frac{65}{50}\right) - 1. \tag{2.197}$$

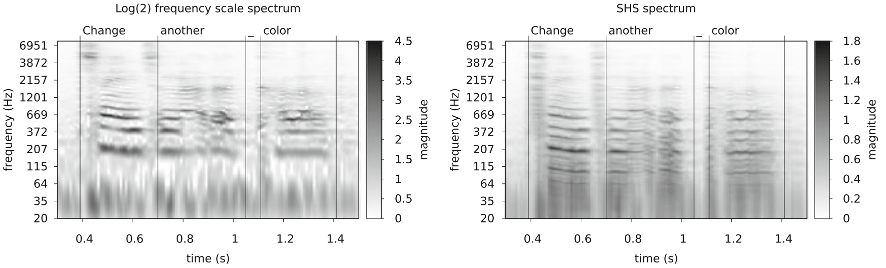

Fig. 2.7 *Left* octave (log(2)) scaled spectrum with peak enhancement, smoothing, and auditory weighting applied ($X_M^{(oct,w)}$); *right* subharmonic sum spectrum X_H. Sample sentence from the AVIC database (Sect. 6.1.3); female speaker, words: "change another colour"

The resulting spectrum $X_M^{(oct,w)}(m^{(oct)}) = W(m^{(oct)})X_M^{(oct)}(m^{(oct)})$ (see Fig. 2.7, left) is then shifted by a constant factor along the octave frequency axis (5), scaled by a factor h_i and all I scaled versions of the spectrum are summed:

$$X_H(m^{(oct)}) = \sum_{i=1}^{I} h_i X_M^{(oct,w)} \left(m^{(oct)} + \log_2 i\right) \tag{2.198}$$

Thereby, I is the number of harmonics which are considered, i.e., the number of harmonics to add up. The factor h_i is given by Hermes (1988) as $h_i = \gamma^{i-1}$ with the compression factor[38] $\gamma = 0.84$. The result $X_H(m)$ is called the subharmonic sum spectrum. In Fig. 2.7 (right) it can be seen that, in the subharmonic sum spectrum the peak corresponding to the pitch (around 200 Hz) is stronger than in the octave scale spectrum (left, in Fig. 2.7) and it is more narrow. A more narrow peak gives a better spectral resolution for pitch estimation. In SHS the spectral resolution in the low-frequency regions of the spectrum is increased by the influence of the higher harmonics.

Hermes (1988) now defines the pitch estimate as the location of the maximum of $X_H(m)$. Further, a method based on sub-frame correlation coefficients is given to perform a voiced/unvoiced decision for each frame. In this thesis a more complex method for picking multiple pitch candidate peaks is applied, and an alternative method for performing voiced/unvoiced decisions directly from the subharmonic sum spectrum is introduced below (Eq. (2.211)). First, all possible pitch candidates are identified:

N_c (typically 5–6) highest peaks are found by iterating through all the local maxima of $X_H(m^{(oct)})$ and retaining the N_c ones with highest magnitude. For each of these peaks the magnitude X_i and the discrete octave scale frequency $m_i^{(oct)}$ are stored. In the next step the exact frequency and magnitude of each peak candidate i is refined by quadratic interpolation from three points. The peak candidates are stored sorted by

[38] γ can be changed in openSMILE via the *compressionFactor* option of the `cPitchShs` component.

their magnitude, not by their frequency. This algorithm is the "greedy" peak picking algorithm, because it finds all N_c candidates with highest magnitude. The non-greedy version (in old versions of openSMILE, and in feature sets up to the INTERSPEECH 2012 Speaker Trait Challenge (IS12) set—see Sect. 3.4) only detected multiple candidates if the candidates with higher frequencies than the first had a greater magnitude than the first.[39] This is acceptable when the best candidate (the first one) is selected directly without Viterbi smoothing (as in the INTERSPEECH 2011 Speaker State Challenge (IS11) set, for example—see Sect. 3.3), otherwise the greedy algorithm should be used. This is implemented in the ComParE feature set (Sect. 3.5).

Quadratic interpolation tries to fit a parabola to three points (x_1, y_1), (x_2, y_2), (x_3, y_3) where the x-values meet the condition $x_1 < x_2 < x_3$ and the y-values satisfy $y_1 < y_2$ and $y_3 < y_2$, i.e., the point (x_2, y_2) represents a local maximum. The interpolation algorithm computes the values of a, b, and c of the quadratic equation:

$$y = ax^2 + bx + c. \tag{2.199}$$

With the three given points a set of three linear equations is constructed, which is solved for the variables a, b, and c. The closed form solution yields the following equations for the parabola parameters:

$$a = \frac{y_1x_2 + y_2x_3 + y_3x_1 - y_3x_2 - y_2x_1 - y_1x_3}{d}, \tag{2.200}$$

$$b = \frac{x_1^2y_2 + x_2^2y_3 + x_3^2y_1 - x_3^2y_2 - x_2^2y_1 - x_1^2y_3}{d}, \tag{2.201}$$

$$c = \frac{x_1^2x_2y_3 + x_2^2x_3y_1 + x_3^2x_1y_2 - x_3^2x_2y_1 - x_2^2x_1y_3 - x_1^2x_3y_2}{d}, \tag{2.202}$$

with the denominator:

$$d = x_1^2x_2 + x_2^2x_3 + x_3^2x_1 - x_3^2x_2 - x_2^2x_1 - x_1^2x_3. \tag{2.203}$$

The refined estimates of the original local maximum in (x_2, y_2) are now defined by the vertex of the parabola given by a, b, and c. By applying elementary calculus (solving for extreme values of functions), the coordinates x_v and y_v of the vertex are found as:

$$x_v = \frac{-b}{2a}, \tag{2.204}$$

$$y_c = c - ax_v^2. \tag{2.205}$$

[39] The greedy peak picking algorithm behaviour is achieved in openSMILE when the `greedyPeakAlgo` option is set to 1. The old (non-greedy) version of the algorithm searched through the peaks from lowest to highest frequency and considered the first peak found as the first candidate. Another candidate was only added if the magnitude was higher than that of the previous first candidate. This behaviour was sub-optimal for Viterbi based smoothing, which requires multiple candidates to evaluate the best path among them.

Here,

$$x_{1,i} = F^{(oct)}\left(m_i^{(oct)} - 1\right), \tag{2.206}$$

$$x_{2,i} = F^{(oct)}\left(m_i^{(oct)}\right), \tag{2.207}$$

$$x_{3,i} = F^{(oct)}\left(m_i^{(oct)} + 1\right), \tag{2.208}$$

where $F^{(oct)}\left(m_i^{(oct)}\right)$ is the octave scale frequency of the bin index of the ith peak candidate. The parabolic refined estimates of the pitch candidate frequency and amplitude are computed from the exact parabola vertex $(x_{v,i}, y_{v,i})$ with the following equations (assuming a $\log_2$ scale):

$$f_i^{(lin)} = 2^{x_{v,i}}, \tag{2.209}$$

$$X_i' = y_{v,i}. \tag{2.210}$$

Next, the arithmetic mean μ_H of $X_H(m^{(oct)})$ is computed. The voicing probability p_v associated with each pitch candidate i is then given as a function of the pitch candidate's refined amplitude X_i':

$$p_{v,i} = 1 - \frac{\mu_H}{X_i'}. \tag{2.211}$$

The above equation is based on the assumption that a subharmonic sum spectrum of a voiced signal will have a higher peak with respect to the mean than that of a noise like signal. Practical experiments have shown that this assumption holds true for many types of speech signals, except for signals with very low energy. Thus, a RMS energy (Eq. (2.41)) threshold is defined as 0.001 (assuming that the input signal $x(n)$ is in the range of $[-1; +1]$). Every frame with a RMS energy below this threshold is assumed to be unvoiced (i.e., $p_{v,i} = 0$ is assigned).[40]

From the N_c pitch candidates, now, a single—most probable—candidate must be selected. A simple solution[41] is to choose the candidate with the highest refined magnitude X_i', i.e., the highest voicing probability (see Fig. 2.8, right). It is better, however, to perform this choice in context with past and future pitch candidates in order to smooth the pitch trajectory and eliminate errors which are evident as rapid jumps. Such a smoothing method is presented in the next section.

[40] In openSMILE this behaviour is not implemented in the `cPitchShs` component, but is rather implemented via the configuration, e.g., for the `smileF0_base.conf` and `IS13_ComParE.conf` configurations. Thereby, the `cValbasedSelector` component is used to force F0 values to 0 (indicating unvoiced parts) if the energy falls below the threshold.

[41] Available in openSMILE via the `cPitchSmoother` component.

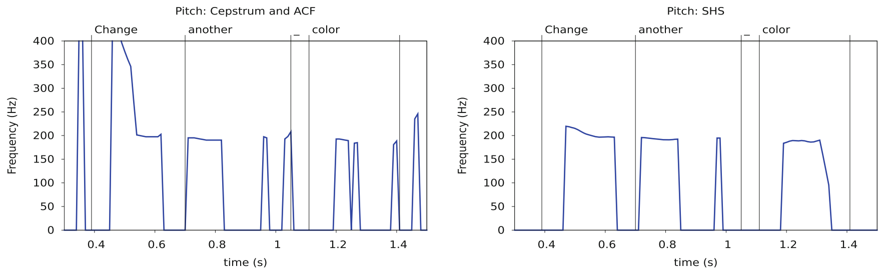

Fig. 2.8 *Left* pitch computed with the Cepstrum/ACF algorithm (Sect. 2.2.11.2); *right* pitch computed with the SHS algorithm (Sect. 2.2.11.4). Sample sentence from the AVIC database (Sect. 6.1.3); female speaker, words: "change another colour"

2.2.11.5 Post Smoothing with Dynamic Programming

To N_c pitch candidates $i = 1 \dots N_c$ and associated scores (magnitudes, voicing probabilities, etc.) estimated by an arbitrary PDA, dynamic programming can be applied to find the best trajectory of pitch values and associated voicing probabilities. In order for the algorithm to also perform the voiced/unvoiced decision dynamically, an additional pitch candidate for the unvoiced case is added ($i = N_c + 1$). The algorithm described by Luengo (2007) is adopted in this thesis.[42] It implements a Viterbi algorithm based least cost search. Thereby each pitch candidate i at time (frame) n has an associated cost value $c_i(n)$. This cost has two components: the *local cost* $c_i^{(l)}(n)$, i.e., a cost value based on the likelihood of the current pitch candidate being a valid candidate, and the *transition cost* $c_i^{(t)}(n)$, i.e., a cost value which considers the current candidate in the context of possible past and future candidates. The total cost is given as: $c_i(n) = c_i^{(l)}(n) + c_i^{(t)}(n)$.

The local cost $c_i^{(l)}(n)$ can be split into two parts (Luengo 2007):

$$c_i^{(l)}(n) = c_i^{(V)}(n) + w_{thr} c_i^{(thr)}(n). \tag{2.212}$$

The first part ($c^{(V)}$) is related to the costs of voiced parts. It is composed of a cost derived from the voicing probability $p_{v,i}(n)$ and a cost derived from the frequency $f_i^{(lin)}(n)$ of the candidate with respect to the expected range of pitch frequencies:

$$c_i^{(V)}(n) = \begin{cases} w_{local}\left(-\log(p_{v,i}(n)) + p_{thr,i}(n)\right) + w_{range} c_i^{(frange)}(n) & i \leq N_c \\ 0 & i = N_c \end{cases}, \tag{2.213}$$

[42] In openSMILE the Viterbi based pitch smoothing is implemented in the `cPitchSmootherViterbi` component.

with the statically defined frequency weighting cost:

$$c_i^{(frange)}(n) = \begin{cases} 2 & \text{if } f_i^{(lin)}(n) = 0 \\ 1 - \frac{f}{100} & \text{if } 0 < f_i^{(lin)}(n) < 100 \\ 0 & \text{if } 100 \leq f_i^{(lin)}(n) < 350 \\ \frac{f-350}{250} & \text{if } 350 \leq f_i^{(lin)}(n) < 600 \\ 1.2 & \text{if } f_i^{(lin)}(n) \geq 600 \end{cases}, \tag{2.214}$$

$$p_{thr,i}(n) = \begin{cases} 0 & p_{v,i}(n) < \theta_v \\ w_{thr} & \text{otherwise} \end{cases}, \tag{2.215}$$

where θ_v is the voicing probability threshold for voiced/unvoiced frames. The threshold depends on the PDA used: for ACF based voicing probabilities, it is typically in the range of 0.4–0.6 (default 0.5), for the SHS algorithm it is typically 0.6–0.8 (default 0.7). As an extension to (Luengo 2007), in this thesis the frequency range weighting cost $c_i^{(frange)}(n)$ was added to the term for voiced candidates ($i \leq N_c$) in order to account for different strengths of the candidates. As improvement, a speaker dependent frequency range cost function $c_i^{(frange)}(n)$ could be constructed basing on the incrementally updated mean and standard deviation of a given speaker. A static case of speaker dependent frequency range post-processing was suggested by Luengo (2007), but is not considered in this thesis. Such an adaptation requires speaker ID knowledge, which was not always present or easily accessible in the studies conducted.

The second part of Eq. (2.212) is related to the cost associated with the voicing probability crossing a pre-defined threshold for voiced/unvoiced:

$$c_i^{(thr)}(n) = \begin{cases} 0 & \forall i \leq N_c : p_{v,i}(n) < \theta_v \\ 1 & \text{otherwise} \end{cases}. \tag{2.216}$$

The transition costs for a transition from frame $n-1$ and pitch candidate j to frame n and pitch candidate i are given for all possible transition categories, i.e., voiced–voiced (V–V), voiced–unvoiced (V–U), unvoiced–voiced (U–V), and unvoiced–unvoiced (U–U) by the following equation (Luengo 2007):

$$c_{i,j}^{(t)}(n, n-1) = \begin{cases} c_{i,j}^{(t,VV)}(n, n-1) & \text{for } V{-}V \\ w_{Tvuv} & \text{for } V{-}U \text{ or } U{-}V \\ w_{Tuu} & \text{for } U{-}U \end{cases}, \tag{2.217}$$

with

$$c_{i,j}^{(t,VV)}(n, n-1) = w_{Tvv} \left| \log \frac{f_i^{(lin)}(n)}{f_j^{(lin)}(n-1)} \right| + w_{Tvvd} \left| \frac{f_i^{(lin)}(n)}{f_j^{(lin)}(n-1)} - \frac{f_j^{(lin)}(n-1)}{f_{j^*}^{(lin)}(n-2)} \right|, \tag{2.218}$$

where j^* is the pitch candidate at time $n-2$ of which the frequency is closest to the frequency of the pitch candidate j at time $n-1$. This second *delta continuity constraint* (weight w_{Tvvd}) has also been introduced here as an extension to (Luengo 2007).

Sensible defaults for the cost weights have been tuned empirically by inspecting the final pitch detection results on various speech signals. These are—as used in the ComParE 2013 feature set: $w_{local} = 2$, $w_{thr} = 4$, $w_{range} = 1$, $w_{Tvv} = 10$, $w_{Tvvd} = 5$, $w_{Tvuv} = 10$, and $w_{Tuu} = 0$. A higher value thereby means a higher weight for the corresponding parameter in the cost function, i.e., penalising transitions of the respective kind, or emphasising local cost parameters such as the correct frequency range and correct voicing probability.

In order to find the least cost path, the costs for all possible paths are evaluated recursively. Thereby, if paths meet in the same node, only the one with lowest cost is kept. For details see (Viterbi 1967). In theory, for some worse cases, the whole sequence of inputs has to be processed before the best path can be chosen. In other, more ideal cases, the paths converge after a few frames and can be reduced to one as described above. In this case the best path up to the meeting point is known. In order to make the algorithm work with a fixed maximum delay in on-line systems, a maximum buffer length l_b is introduced. If paths have not merged to a single path at time $n - l_b$, the path with the lowest cost at time $n - l_b$ is selected as the winning path (potentially making a small error) and all other paths are discarded.

2.2.12 F_0 Harmonics

A voiced speech signal is composed of a sinusoid at the fundamental frequency and a series of sinusoids at its harmonics, as well as some noise signal. In the linear speech production model, the fundamental frequency is represented by a Dirac pulse series, which corresponds to a comb spectrum with peaks at integer multiples of F_0. In theory, in the source spectrum all peaks have the same amplitude. The envelope of the comb spectrum is then shaped by the dynamic transfer function of the vocal tract to the spectrum of specific vowels. Thus, the amplitudes of the F_0 harmonics do contain valuable information about the state of the vocal tract. This information is also reflected to some degree in other features such as spectral LLDs (Sect. 2.2.4) or Cepstral LLDs (Sect. 2.2.10). A more direct approach to encoding harmonics information, on the other hand, might reveal additional information and—most important—produces parameters which can be assigned a direct physical meaning, thus making the interpretation of results easier. An example is the ratio of the first harmonic (H_1) to the highest harmonic in the third formant range (A_3) as reported on by Hanson and Chuang (1999) and Hanson (1997).

First, all harmonics need to be identified. The algorithm implemented in openSMILE for this thesis performs the following steps to find the first I harmonics[43]:

1. Starting at the estimated location of the ith harmonic ($i = 0 \dots I$), which is the closest magnitude spectrum bin to the frequency $i \cdot F_0$, the closest local maximum is found in the range $[(i - 0.5) \cdot F_0; (i + 0.5) \cdot F_0]$. The amplitude of this maximum is used as the amplitude H_i of the ith harmonic. If no local maximum is found within the search range, the ith harmonic is marked as a non-existant/invalid harmonic,
2. Duplicates, i.e., where the frequency of the ith harmonic is equal to the frequency of harmonic $i + 1$ are removed by marking the harmonic $i + 1$ as an invalid harmonic,
3. The amplitudes H_i are converted to logarithmic relative amplitudes, by normalising with the amplitude of the 0th harmonic (i.e., amplitude of the F_0 component):

$$H_i^{(rel,\,dB)} = 20 \log_{10}\left(\frac{H_i}{H_0}\right). \tag{2.219}$$

Next, harmonic amplitude differences (ratios in the linear scale) are defined as follows:

$$H_{i-j} = H_i^{(dB)} - H_j^{(dB)}. \tag{2.220}$$

Formant amplitudes A_i are estimated by finding the highest spectral peak (local maximum) in the range $[0.8 \cdot F_i; 1.2 \cdot F_i]$ around the formant centre frequency F_i (estimated via LPC analysis, for example).

2.2.13 Voice Quality

In contrast to prosodic features, such as energy, loudness, and pitch, which describe the "global" melody and intensity of a speech utterance, voice quality features describe micro-prosodic variations within short-time frames (Schuller 2013).

2.2.13.1 Jitter

Jitter (cf. Schuller 2013) describes the variation of the length of the fundamental period from one single period to the next. The length of the first period $n' - 1$ is $T_0(n' - 1)$ and the length of the second period n' is $T_0(n')$. Then, the absolute period to period jitter, also referred to as absolute local jitter, is given as (Schuller 2013):

$$J_{pp}(n') = \left|T_0(n') - T_0(n' - 1)\right| \text{ for } n' > 1 \tag{2.221}$$

[43] In openSMILE version 2.0 and above, these parameters are implemented by the `cHarmonics` component.

This definition yields one value for J_{pp} for every pitch period, except the first one. Often, however, short-time analysis is used, and descriptor values are expected at the rate of the short-time frames (typically 10 ms). Thus, the average Jitter per frame is used in this thesis. For N' pitch periods $n' = 1 \ldots N'$ within one analysis frame the average local Jitter is given as:

$$\overline{J_{pp}} = \frac{1}{N'-1} \sum_{n'=2}^{N'} \left| T_0(n') - T_0(n'-1) \right|. \tag{2.222}$$

In order to make the Jitter value independent of the underlying pitch period length, it is normalised by the average pitch period length. This yields the average relative Jitter[44]:

$$\overline{J_{pp,rel}} = \frac{\frac{1}{N'-1} \sum_{n'=2}^{N'} \left| T_0(n') - T_0(n'-1) \right|}{\frac{1}{N'} \sum_{n'=1}^{N'} T_0(n')}. \tag{2.223}$$

The variance of Jitter across frames can be described via the "Jitter of the Jitter", i.e., the delta period to period Jitter J_{ddp}. It is defined as the difference between two consecutive J_{pp} values:

$$J_{ddp}(n') = \left| J_{pp}(n') - J_{pp}(n'-1) \right| \text{ for } n' > 2. \tag{2.224}$$

The delta Jitter is also expressed as average over a short-time frame, and is normalised by the average period length[45]:

$$\overline{J_{ddp,rel}} = \frac{\frac{1}{N'-2} \sum_{n'=3}^{N'} \left| \left| T_0(n') - T_0(n'-1) \right| - \left| T_0(n'-1) - T_0(n'-2) \right| \right|}{\frac{1}{N'} \sum_{n'=1}^{N'} T_0(n')}. \tag{2.225}$$

For local Jitter at least two, and for delta Jitter at least three pitch periods per analysis frame are required. Since for low pitch (e.g., male voices) this might not always be the case for typical analysis frame lengths (20–50 ms), in the implementation developed for this thesis, the last frame period length and the last local Jitter from the previous frame are stored for use on the next frame. Further, the last audio samples x of the previous frame, which belong to an incomplete period, are also stored and appended to the beginning of the next frame. The stored values are all set to 0 or deleted (in case of the previous audio samples) when an unvoiced frame is encountered.

[44] This definition of Jitter is implemented in openSMILE in the `cPitchJitter` component. It can be enabled via the `jitterLocal` option.

[45] This definition of delta Jitter is implemented in openSMILE in the `cPitchJitter` component. It can be enabled via the `jitterDDP` option.

In order to determine the exact length of pitch periods, a correlation based waveform matching algorithm is implemented. An extended and modified version of the basic matching algorithm by Boersma (2001) is implemented. The algorithm is guided by a frame based estimate of $T_0 = 1/F_0$ determined by a spectral domain PDA as described in Sect. 2.2.11. The estimate of T_0 is used to limit the range of the period cross-correlation to improve both the robustness against noise and computation time. The waveform matching algorithm operates directly on unwindowed frames of the speech signal $x(n)$.

The algorithm performs the following three steps for voiced frames:
(a) A range is defined for the allowed pitch period length in samples, based on the estimated frame T_0 and the sampling rate T_s:

$$\begin{aligned} N^{(T)}_{0,\,min} &= (1-\alpha)\,\frac{T_0}{T_s}, \\ N^{(T)}_{0,\,max} &= (1+\alpha)\,\frac{T_0}{T_s}, \end{aligned} \tag{2.226}$$

where α is the relative search range[46] for the correlation from 0.01–0.99, typically 0.1–0.25.
(b) For each integer (sample) period length $N^{(T)} = N^{(T)}_{0,\,min} \ldots N^{(T)}_{0,\,max}$, the normalised cross-correlations (Pearson Correlation Coefficient—cf. Eq. (5.8)) of two periods ranging from a start index n to $(n + N^{(T)} - 1)$ and from $(n + N^{(T)})$ to $(n + 2N^{(T)} - 1)$ are computed and stored in a vector $\underline{c}$. The highest local maximum of $\underline{c}$ is now found, parabolic interpolation is applied (Eq. (2.199) in Sect. 2.2.11.4) to enhance the peak, and the resulting period length for the first period is computed from the (real-valued) period length $N^{(T)'}$ corresponding to the interpolated peak location. The corresponding period length in seconds is given as $T_0 = N^{(T)'} T_s$. The amplitude $CC = \underline{c}(N^{(T)'})$ of the correlation peak is used to verify the voicedness of the signal[47]: only if the amplitude is higher than 0.5, the current period is considered for the average Jitter computation (Eqs. (2.223) and (2.225)).

The average, minimum, and maximum CC for all periods within a frame constitute further acoustic descriptors[48]: they describe the variation from period to period, i.e., the regularity of speech signal periods. The average CC will be higher for a regular voice signal. It will be low for a voice signal with laryngalisation (irregular phonation; creaky voice—cf. Schuller and Batliner 2013).
(c) algorithm step (b) is repeated for the next two periods, starting at the new start index $n_1 = n + N^{(T)}$ (end of the previous period), until the end of the analysis frame is reached. The correlation search range defined by $N^{(T)}_{0,\,min} \ldots N^{(T)}_{0,\,max}$ is kept constant for the whole frame, i.e., *not* updated on a period to period base.

[46] `searchRangeRel` option of the `cPitchJitter` component in openSMILE.

[47] `minCC` option in openSMILE.

[48] `sourceQualityMean` and `sourceQualityRange` options in `cPitchJitter` of openSMILE.

It is to note at this point, that the waveform matching algorithm does not estimate the phase of the periods. The beginning of the first period in the first voiced frame is chosen randomly as the first sample in that frame. If a correct tracking of all periods in the voiced segment is assumed, the phase should be constant throughout the voiced segment, though.

2.2.13.2 Shimmer

Shimmer describes amplitude variations of consecutive voice signal periods (Boersma 2001).

The waveform matching algorithm described in Sect. 2.2.13.1 is used to identify the exact period boundaries. As the phase of the segments determined with this algorithm is random, the maximum and minimum amplitude ($x_{max,n'}$ and $x_{min,n'}$) within each period are found. In analogy to Jitter, the period to period amplitude Shimmer is then expressed as (Schuller 2013):

$$S_{pp}(n') = \left|A(n') - A(n'-1)\right|, \tag{2.227}$$

with the peak to peak amplitude $A(n') = x_{max,n'} - x_{min,n'}$.

This type of Jitter is very sensitive to additive noise, as this introduces large, random peak amplitude variations which are not related to the original speech signal. To mitigate this effect, the signal $x(n)$ can be low-pass filtered with a cut-off frequency between 600 and 800 Hz. This roughly preserves the first and—partially—the second formant, and high-pass filtered with a cut-off of 50 Hz to remove low frequency noise and slowly varying static offsets which might be introduced by low quality recording equipment. Alternatively, as implemented in this thesis, the period to period RMS amplitude variation can be used. Thereby the RMS amplitude (=unnormalised RMS energy, cf. Eq. (2.41)—without normalising by the frame length) $E_{rmsU,n'}$ is computed for each period n':

$$E_{rmsU,n'} = \sqrt{\sum_{n=0}^{N-1} x^2(n)}, \tag{2.228}$$

where N is the period length $N^{(T)}$. The period to period RMS Shimmer is then expressed as:

$$S_{pp,rms}(n') = \left|E_{rmsU,n'} - E_{rmsU,n'-1}\right|. \tag{2.229}$$

Delta Shimmer, like delta Jitter, can also be computed by replacing Jitter with Shimmer in Eq. (2.224).

As for Jitter, the period to period Shimmer values are averaged over the scope of a short-time frame in order to synchronise the rate of this descriptor with the constant rate of all other short-time descriptors. The averaged, relative Shimmer values are referred to as $\overline{S_{pp(,rms)}}$ for the local Shimmer and $\overline{S_{ddp(,rms)}}$ for the delta Shimmer. They are expressed as amplitude ratios, i.e., the per period amplitude values are normalised to the per frame average amplitude (peak or rms, respectively):

$$\overline{S_{pp(,rms),rel}} = \frac{\frac{1}{N'-1}\sum_{n'=2}^{N'} S_{pp(,rms)}(n')}{\frac{1}{N}\sum_{n'=1}^{N'} A(n')}, \tag{2.230}$$

$$\overline{S_{ddp(,rms),rel}} = \frac{\frac{1}{N'-1}\sum_{n'=3}^{N'} S_{ddp(,rms)}(n')}{\frac{1}{N}\sum_{n'=1}^{N'} A(n')}, \tag{2.231}$$

where the peak to peak amplitude $A(n')$ has to be substituted by the RMS amplitude in case of the $\overline{S_{*,rms,rel}}$ versions.

Amplitude ratios in audio signal processing are commonly expressed on a logarithmic scale, i.e., in dB. For converting the Shimmer ratios to dB the following equation is implemented:

$$S_{dB} = \begin{cases} 20\log_{10}(S_{rel}) \text{ dB} & S_{rel} > 10^{-50} \\ -1000 & \text{otherwise} \end{cases}. \tag{2.232}$$

2.2.13.3 Harmonics-to-Noise Ratio (HNR)

The HNR is defined as the ratio of the energy of harmonic signal components to the energy of noise like signal components. In an early study by Yumoto and Gould (1981), it is referred to as H/N ratio, and an algorithm for computing the ratio in the time domain is given: a sequence of N' periods with the same period length is considered and the average period waveform is computed. The average period waveform is assumed to strongly reflect the harmonic components because these are assumed to be constant over the N' periods, while the noise is uncorrelated and is theoretically cancelled out by the averaging. The noise energy is then computed by subtracting the average waveform from each individual waveform and computing the RMS energy E_{noise} of the remaining signal over all N' periods. The harmonic energy E_{harm} is computed as RMS energy of the average waveform. The HNR is then expressed as HNR_{wf}:

$$HNR_{wf} = \frac{E_{harm}}{E_{noise}}, \tag{2.233}$$

where the subscript wf denotes the direct waveform method. In this thesis, the above algorithm is implemented for voice quality analysis based on the waveform matching algorithm in Sect. 2.2.13.1. The average waveform is thereby computed over the N' fundamental periods in a short-time frame. The advantage of the method is that under ideal conditions (no jitter and shimmer, white noise) the result will be very accurate. However, ideal conditions do not exist for real world speech signals, and, moreover, the algorithm is vulnerable to errors from the pitch period detection process.

Another method for computing the HNR, which does not require individual pitch periods in the time domain, is based on the ACF. The method is similar to the estimation of the voicing probability as used in ACF based PDAs (cf. Eq. (2.191)). The HNR is given as HNR_{acf} (cf. Schuller 2006):

$$HNR_{acf} = \frac{ACF_{T_0}}{ACF_0 - ACF_{T_0}}, \tag{2.234}$$

where ACF_{T_0} is the amplitude of the autocorrelation peak at the fundamental period (see Sect. 2.2.11.2) and ACF_0 is the 0th ACF coefficient (equivalent to the quadratic frame energy).

Since the HNR is an energy ratio, it is best expressed on a logarithmic scale in dB (Schuller 2013) as $HNR_{wf,log}$ and $HNR_{acf,log}$:

$$HNR_{wf,log} = 20 \log_{10} \left(HNR_{wf}\right) \text{dB} \tag{2.235}$$

$$HNR_{acf,log} = 10 \log_{10} \left(HNR_{acf}\right) \text{dB} \tag{2.236}$$

Both logarithmic HNR values are floored to -100 dB.

An alternative method for computing the HNR is given by Krom (1993), for example. It exploits the Cepstral domain for separating the harmonic and noise energy components by applying comb-liftering.

2.2.14 *Tonal Features*

Tonal features are spectral features which are related to music theory. The tonal features which are described in this section are based on the 12-tone scale of western popular music. The features were originally introduced by Fujishima (1999) as Pitch Class Profiles (PCP) for the purpose of automatic chord recognition. They were later also referred to as CHROMA features because they reflect the "colour", i.e., the tonal shape, of the spectrum (Müller 2007). Moreover, PCP were improved by several tweaks and extensions, such as the post-processing method CHROMA Energy-distribution Normalised Statistics (CENS) suggested by Müller et al. (2005b) (Sect. 2.2.14.3).

2.2.14.1 Semitone Spectrum

The first step in the computation of PCP is to obtain a magnitude spectrum $X_M(m)$ (Schuller 2013) or power spectrum $X_P(m)$ (Fujishima 1999)—usually via a FFT. Then, this spectrum is mapped to an octave scale with 12 semitones per octave (cf. Sect. 2.2.3.3) and bins are combined to one bin per semitone, resulting in a semitone band spectrum (Sect. 2.2.3.4). Alternatively, a semitone band spectrum

$X_M^{(oct)}(b)$ can be computed directly from the linear scale magnitude spectrum—i.e., without mapping to a semitone frequency scale—as described in Sect. 2.2.3.4.

For the semitone band spectrum a first note with a positive non-zero frequency has to be defined on which the scale is built. Typically the note **A** with 110 or 55 Hz is used in this context. The range of the spectrum typically covers 6–8 octaves (72–96 semitones), which—when the first note is an **A** with 55 Hz—corresponds to a frequency of 3.52, 7.04, and 14.08 kHz for the last note on the scale, respectively for 6, 7, or 8 octaves.

The semitone band spectra used in this thesis are implemented via a spectral domain filterbank with rectangular filters, as described in Sect. 2.2.3.4. Variations exist where only peaks in the magnitude spectrum are considered and all other bins are set to 0 (cf. Schuller 2013, p. 64). Such variations are not considered here.

2.2.14.2 Pitch Class Profiles

The goal of PCP is to describe the tonality of a piece of music independent of the actual pitch. These descriptors are also known as CHROMA features (Müller 2007).[49] In the most basic case, for example, the semitone spectrum is warped to a single octave resulting in a 12-dimensional feature vector regardless of how many octaves the semitone spectrum spans.

The mapping of a semitone band spectrum $X_M^{(oct)}(b)$ to a PCP vector $PCP(s)$ with $s = 1..S$, and S being the number of semitones to warp to, is given by the following equation:

$$PCP(s) = \frac{1}{O}\sum_{i=0}^{O-1} X_M^{(oct)}(s + i \cdot S), \tag{2.237}$$

where O is the number of octaves to warp.

In general, PCP are not constrained to a 12 dimensional vector. Gómez (2006) describes Harmonic Pitch Class Profiles (HPCP), an extended version of the original PCP, where sampling at sub-semitone level (half or third semitone) is possible, resulting in 24 or 36 dimensional vectors. Also, warping not to a single octave but to two or more consecutive octaves is possible (Schuller 2013), yielding 24 or 36 dimensional feature vectors with semi-tone resolution.

In order to make the PCP features independent of the sound level, each PCP feature vector can be normalised to a length of one:

$$PCP'(s) = \frac{PCP(s)}{\sqrt{\sum_{s=0}^{S-1} PCP(s)^2}}. \tag{2.238}$$

[49] In openSMILE CHROMA features are supported by the `cChroma` component, which requires a semi-tone band spectrum as input, which can be generated by the `cTonespec` component (preferred) or by the (more general) `cMelspec` component.

This normalisation, however, emphasises noise, when the frame has very low energy (Schuller 2013). To avoid problems in this respect, the PCP values should be set to 0 when the frame energy falls below a (low) pre-defined threshold.

2.2.14.3 CENS

PCP features are frame-wise descriptors and thus do not consider context of left and right frames. However, in music rate of change of the tonal structure (240 Beats per Minute (BPM), for example, equals 4 Hz) is well below the frame rate (100 Hz). Thus, it is beneficial to consider the context of neighbouring frames. Above that, noise from non-tonal signal parts (percussion, unvoiced vocals, etc.) might cause differences between subsequent PCP vectors which are not related to tonal changes. CENS features (Müller et al. 2005a) attempt to compensate for these two deficiencies of PCP features by implementing a smoothing over time and a quantisation of the PCP values in order to improve robustness against non-tonal influences.[50]

The quantisation of the PCP amplitude $a = PCP(s)$ is defined as (Schuller 2013):

$$Q(a) = \begin{cases} 4 & 0.4 \leq a \leq 1 \\ 3 & 0.2 \leq a \leq 0.4 \\ 2 & 0.1 \leq a \leq 0.2 \\ 1 & 0.05 \leq a \leq 0.1 \\ 0 & 0 \leq a \leq 0.05 \end{cases} \tag{2.239}$$

Next, a smoothing is applied by convolving the quantised PCP frames $PCP_q(s)$ with a Hanning window of length 11 (Schuller 2013). To reduce redundant data, according to Schuller (2013) this is followed by a downsampling of factor 4. However, in order to be able to process all features at the same rate in the on-line system presented in this thesis, the downsampling is *not* applied here.

2.2.15 *Non-linear Vocal Tract Model Features*

In contrast to the linear model of speech production, recently, the modelling of non-linearities in the human speech production system has gained interest. Especially studies on detection of stress from the voice have investigated methods to model non-linearities in the glottal airflow (Zhou et al. 2001; Zuo and Fung 2011).

[50] In openSMILE CENS features can be computed from CHROMA (PCP) features with the `cCens` component.

2.2.15.1 Critical Band Filterbanks

Instead of approximating the frequency sensitivity and masking of the human auditory system in the frequency domain (as in Sect. 2.2.9.1, for example), a critical band filterbank is constructed in the time domain. In this way, the filterbank of the human auditory system can be approximated very closely with all its non-linear aspects.

In analogy to the frequency domain triangular filter banks for Bark- or Mel-scales (Sect. 2.2.3.4), the centre frequencies of the filters are approximately equidistant on a Bark scale, and the bandwidths are approximately constant there. In this thesis the filterbank from (Zhou et al. 2001) has been adopted. The centre frequencies and bandwidths of the filters are given in Table 2.1. Zhou et al. (2001) show the centre frequencies for bands 1–16. Here, the table was extended for the full auditory

Table 2.1 Critical band filter bank according to Zhou et al. (2001) (up to band 16, bands 17–24 calculated for a filter spacing of 1 Bark, see text for details); the bandwidths (f_{bw}) and linear scale centre frequencies ($f_c^{(lin)}$) are rounded to the closest multiple of 10

Band #	$f_c^{(lin)}$ (Hz)	$f_c^{(bark)}$ (Bark)	f_{bw} (Hz)
1	150	1.5	100
2	250	2.5	100
3	350	3.5	100
4	450	4.5	110
5	570	5.5	120
6	700	6.5	140
7	840	7.5	150
8	1000	8.5	160
9	1170	9.5	190
10	1370	10.5	210
11	1600	11.5	240
12	1850	12.5	280
13	2150	13.5	320
14	2500	14.5	380
15	2900	15.5	450
16	3400	16.5	550
17	4050	17.5	680
18	4800	18.5	870
19	5800	19.5	1150
20	7000	20.5	1290
21	8500	21.5	1710
22	10,500	22.5	2450
23	13,500	23.5	3800
24	18,500	24.5	6670

frequency range up to band 24, assuming a spacing of the centre frequencies $f_c^{(bark)}$ of 1 Bark and a bandwidth f_{bw} as is given by the following equation:

$$f_{bw} = \Theta_{bark}^{-1}(f_c^{(bark)} + 0.5) - \Theta_{bark}^{-1}(f_c^{(bark)} - 0.5). \tag{2.240}$$

Two types of filters are found in the literature (cf. e.g., Zhou et al. 2001) for the implementation of critical band filterbanks: gammatone filters and Gabor filters, both as introduced in Sect. 2.2.3.5.

All filters in this thesis are implemented as FIR filters for stability and implementation reasons. In order to implement filters which are described by an infinite impulse response with FIR filters, the impulse response needs to be truncated. This introduces a small error compared to the ideal IIR filter. The amount of error can be controlled by setting the length of the truncated impulse response. A longer impulse response will lead to a smaller error.

2.2.15.2 Teager Energy Operator Envelopes

The Teager energy operator (TEO) is an energy operator where a correction term is added to the quadratic energy operator (Sect. 2.2.2). It is also known as the Teager-Kaiser energy operator because it was introduced in papers by Kaiser (1993). This term accounts for non-linearities which are supposed to occur in the human vocal tract (Teager and Teager 1990).

The TEO $\Phi\{x(n)\}$ is defined for a discrete time signal $x(n)$ as (Kaiser 1993):

$$\Phi\{x(n)\} = x^2(n) - x(n+1)x(n-1). \tag{2.241}$$

This TEO is now applied to each of the filterbank output signals $x_b(n)$ (b is the band number; see previous section) and the envelopes of the resulting signals $x_b^{(\Phi)}(n)$ are computed, i.e., by computing the RMS energy for short-time frames or by applying a maximum operator to each short-time frame.

2.3 Derived Features and Post-processing of Low-Level Descriptors

Almost all of the LLDs shown in the previous sections are computed from isolated frames of audio. No context from previous or future frames is considered, i.e., the descriptors do not capture any signal dynamics beyond the frame boundaries.

To mitigate this issue, post-processing of the LLD signals is performed and derived features are computed. The post-processing implemented for this thesis consists of moving average smoothing as suggested by Schuller (2006) (see Sect. 2.3.4). The derived features consist of jth order simple differences and jth order delta regression coefficients (Sect. 2.3.1 and Sect. 2.3.2, respectively).

Other, more specific, variants of derived features have already been described along with the respective LLDs, such as the spectral differences (Sect. 2.2.4.11) or the CENS features (Sect. 2.2.14.3), for example. In a strict sense, these should not be named LLDs, but derived descriptors. However, as they are so specific to a certain descriptor (esp. CENS), they will be herein treated as LLDs nonetheless even though they have been computed from multiple subsequent frames.

2.3.1 Differences

The most simple derived feature which captures signal dynamics beyond a single frame, is the first order simple difference function.[51] It corresponds to the first order differential $\frac{\delta x}{\delta t}$ of a continuous signal $x(t)$. For a discrete signal $x(n)$, which describes the value of an arbitrary LLD over the frame index n, the simple difference function $d_1(n)$ is expressed as:

$$d_1(n) = x(n) - x(n-1) \tag{2.242}$$

The above function is in principle only defined for $n \geq 1$, if $x(n)$ is defined for $n \geq 0$. In practical implementations it is desirable, however, to obtain one vector of derived features for every vector of LLDs, which especially concerns the first frame. Thus, the signal $x(n)$ is padded for negative $x(n)$ (here: up to $n = -1$) either with zeros or with the value of $x(0)$. Padding with zeros is unsuitable for difference features because the difference $d_1(0)$ will be very large compared to the subsequent differences, if the magnitude of $x(n)$ is rather large. In this thesis, therefore, the padding with the value of $x(0)$ is implemented, i.e., $d_1(0) = 0$ always.

2.3.2 Delta Regression Coefficients

A different approach to the difference function is suggested by Young et al. (2006).[52] Thereby, a context window of length W is defined, over which the difference function is computed and smoothed according to the following regression formula:

$$\delta_1^W(n) = \frac{\sum_{i=1}^{W} i \cdot (x(n+i) - x(n-i))}{2\sum_{i=1}^{W} i^2} \tag{2.243}$$

[51] In openSMILE the simple difference function can be applied with the `cDeltaRegression` component with the delta window size set to 0 (option `deltaWin` = 0).

[52] In openSMILE these delta regression coefficients can be computed with the `cDeltaRegression` component.

Hence, these coefficients $\delta_1^W(n)$ are called *delta regression coefficients*. The size of the context window[53] determines the amount of context—and with that the amount of smoothing—that is considered in the computation of the difference around the pivot point at n. When using a larger window, the coefficients will reflect more the mid-term and long-term dynamics of the signal, while with a shorter window, the coefficients will capture the short-term dynamics. Thus, in theory, it makes sense to use delta regression coefficients with different window sizes in the same feature set, although this has not yet been considered in this thesis. The default window size suggested by Young et al. (2006) of $W = 2$ is adopted here.

2.3.3 *Higher Order Delta Regression Coefficients and Differences*

The simple difference and the delta regression coefficients shown in the previous two sections can be extended to higher order differences, which capture higher order dynamics of the signal. These correspond to higher order (j) differentials $\frac{\delta^j x}{\delta t}$ of the continuous signal $x(t)$. The second order difference, for example, is commonly known as acceleration coefficient (from physics, where the first order differential of the distance as a function of time represents speed and the second order differential represents acceleration).

The jth order discrete difference (d) or delta regression (δ) function is expressed recursively as the difference/delta of the ($j-1$)th order difference/delta function (Young et al. 2006):

$$d_j(n) = d_{j-1}(n) - d_{j-1}(n-1), \tag{2.244}$$

$$\delta_j^W(n) = \frac{\sum_{i=1}^{W} i \cdot \left(\delta_{j-1}^W(n+i) - \delta_{j-1}^W(n-i)\right)}{2\sum_{i=1}^{W} i^2}. \tag{2.245}$$

2.3.4 *Temporal Smoothing*

The short-time analysis (Sect. 2.1.3) creates artefacts, which can be reduced by averaging descriptors over a small number of neighbouring frames.[54] Schuller (2006) suggests the use of a moving average filter with a window length of $W = 3$. The filter can be expressed (for odd W only) as:

[53] Option `deltaWin` in openSMILE component `cDeltaRegression`.

[54] In openSMILE the smoothing via a moving average window is implemented in the `cContourSmoother` component. Feature names often carry the suffix `_sma`, which stands for 'smoothed (with) moving average (filtering)'.

$$x_{sma}(n) = \frac{1}{W} \sum_{i=-(W-1)/2}^{(W-1)/2} x(n+i), \tag{2.246}$$

where $x_{sma}(n)$ is the smoothed output (*sma* stands for 'smoothed (with) moving average (filtering)'). A reasonable default for removing windowing artefacts is $W = 3$.

2.4 Supra-Segmental Features

In contrast to ASR which deals with short-term phenomena such as phonemes and words, other speech analysis tasks such as affect recognition, speaker state and trait analysis, and music analysis tasks such as mood recognition or chord and key recognition deal with rather long-term phenomena. E.g., the affective state of a person does not change every second, rather even a few seconds of speech material are required in most cases (even by humans) to asses the emotional state of a person reliably.

To enable machines to robustly model such long-term phenomena, either the classification framework must be able to handle long-range temporal dependencies between inputs (Sect. 2.5.2.2), or the features must summarise information over a meaningful unit of time (where the length depends on the analysed phenomenon). The latter approach has been mainly followed in this thesis and it was shown to be highly effective (cf. e.g., Schuller 2013; Schuller and Batliner 2013 and Chap. 6).

The following sections will show various mechanisms to summarise features over a segment of given length. Two basic categories are distinguished hereby:

1. methods which map a segment of a fixed length to a feature vector of a fixed length, where the length of the resulting vector is proportional to the length of the segment, and
2. methods which map a segment of variable length to a feature vector of a fixed length.

2.4.1 Stacking of Low-Level Descriptors

Probably the most straightforward approach to generate a single feature vector $\underline{X}$ from a series of N LLD feature vectors $\underline{x}(n)$ with $n = 0 \ldots N-1$ is to stack all the LLD vectors to a single, large vector:

$$\underline{X} = \begin{bmatrix} \underline{x}(0) \\ \underline{x}(1) \\ \ldots \\ \underline{x}(N-1) \end{bmatrix} \tag{2.247}$$

In $\underline{X}$ all the information of the original features is contained. However, the size of $\underline{X}$ is proportional to the length N of the series of LLD vectors. If the dimensionality of $\underline{x}(n)$ and N are large, the dimensionality of $\underline{X}$ grows rapidly, which makes this approach unsuitable for high dimensional LLD vectors and long segments.

This approach can be used for the sub-second segments and a dimensionality of the LLD vector of approximately 50 or smaller. It cannot be used to summarise LLDs in segments of variable length.

2.4.2 Statistical Functionals

In order to handle segments with variable length and get rid of the dependency of the feature vector dimensionality on the segment length, statistical functionals can be applied to the time series of LLDs. A functional $\mathscr{F}$ maps a series of values $x(n)$ to a single value $X_{\mathscr{F}}$ (Schuller 2013):

$$x(n) \overset{\mathscr{F}}{\to} X_{\mathscr{F}} \tag{2.248}$$

Thus, the result is independent of the length of the input. Examples of commonly used functionals are the arithmetic mean, standard deviation, maximum value, and the minimum value. Typically these functionals are applied to each LLD individually, i.e., they are referred to as *univariate functionals*[55] by the author of this thesis. Functionals can also be applied to multiple descriptors at the same time, such as the covariance or correlation between two descriptors. Such functionals are named *multivariate functionals*. However, they are beyond the scope of this thesis and not considered here.

The following sections give an overview on the most frequently used functionals for audio analysis and describe all the functionals used in this thesis in detail. A univariate time series will be denoted by $x(n)$. Each time series has N elements $n = 0 \ldots N - 1$.

2.4.2.1 Means

Various types of mean values are implemented.[56] The most common one, often referred to simply as *mean* μ, is the arithmetic mean μ_a:

[55] In openSMILE univariate functionals are accessible via the `cFunctionals` component.

[56] Implementations of mean value related functionals are contained in the `cFunctionalMeans` component in openSMILE, which can be activated by setting `functionalsEnabled = Means` in the configuration of `cFunctionals`.

$$\mu_a = \frac{1}{N}\sum_{n=0}^{N-1} x(n) \tag{2.249}$$

Variations are the arithmetic mean of absolute values $\mu_{|a|}$:

$$\mu_{|a|} = \frac{1}{N}\sum_{n=0}^{N-1} |x(n)|\,, \tag{2.250}$$

and the arithmetic mean of positive (μ_{a+}) or negative (μ_{a-}) values:

$$\mu_{a+} = \frac{1}{N_+}\sum_{n=0}^{N-1} x(n) \cdot s_p\left(x(n)\right), \tag{2.251}$$

$$\mu_{a-} = \frac{1}{N_-}\sum_{n=0}^{N-1} x(n) \cdot s_n\left(x(n)\right), \tag{2.252}$$

where N_+ and N_- are the counts of positive and negative values in $x(n)$, respectively. s_p and s_n mask out non-positive and non-negative values, respectively:

$$s_p\left(x(n)\right) = \begin{cases} 0 & x(n) \leq 0 \\ 1 & x(n) > 0 \end{cases}, \tag{2.253}$$

$$s_n\left(x(n)\right) = \begin{cases} 0 & x(n) \geq 0 \\ 1 & x(n) < 0 \end{cases}. \tag{2.254}$$

Other types of means implemented, are the quadratic mean μ_q:

$$\mu_q = \frac{1}{N}\sum_{n=0}^{N-1} x(n)^2, \tag{2.255}$$

the root-quadratic mean μ_{rq}:

$$\mu_{rq} = \sqrt{\frac{1}{N}\sum_{n=0}^{N-1} x(n)^2}, \tag{2.256}$$

and the geometric mean μ_g:

$$\mu_g = \sqrt[N]{\prod_{n=0}^{N-1} |x(n)|}. \tag{2.257}$$

The above equation for the geometric mean is numerically inefficient for large N. A more efficient method—which is used in this thesis[57]—operates in the logarithmic domain:

$$\mu_g = \exp\left(\frac{1}{N}\sum_{n=0}^{N-1} \log|x(n)|\right). \tag{2.258}$$

The geometric mean is only defined, if all $x(n) \overset{!}{=} 0$. Thus, zero-values are excluded in the computation of the geometric mean, i.e., μ_g in this thesis always refers to the geometric mean of all $x(n) \overset{!}{=} 0$.

Further, for some LLDs (e.g., pitch) it might be of high relevance to compute other types of mean values only from non-zero values. Therefore, the superscript $^{(nz)}$ shall denote a non-zero mean, i.e., a mean value computed only from values $x(n) \overset{!}{=} 0$. For instance, $\mu_a^{(nz)}$, $\mu_{|a|}^{(nz)}$, $\mu_q^{(nz)}$, and $\mu_{rq}^{(nz)}$. By the definition in the last paragraph, the geometric mean is always computed from non-zero values only, thus $\mu_g = \mu_g^{(nz)}$. In the same way as for the arithmetic mean, the quadratic mean and the root-quadratic mean can be computed only for positive and negative values, denoted by $\mu_{(r)q+}$ and $\mu_{(r)q-}$.

Not a mean value, but related to the non-zero mean values[58] (thus listed in this section), is the number of non-zero values N_{nz} in $x(n)$. This number can be normalised to the total number of values in the series (N)[59]: $N_{nz,rel} = N_{nz}/N$.

Further, the *flatness* μ_f of a time series is described by the ratio of the geometric mean to the arithmetic mean (of absolute values in both cases):

$$\mu_f = \begin{cases} \dfrac{\mu_g}{\mu_{|a|}} & \mu_{|a|} > 0 \\ 1 & \mu_{|a|} = 0 \end{cases} \tag{2.259}$$

This measure has originated as spectral flatness LLD, where it was applied to power spectra (cf. Sect. 2.2.4.5). Here it is generalised as a mean related functional which can be applied to any time series $x(n)$.

2.4.2.2 Moments

The arithmetic mean, as described in the previous section is also known as the first order statistical moment. Higher order statistical moments[60] are also very important functionals, in particular the variance and standard deviation (Ververidis and Kotropoulos 2006; Schuller 2006, 2013).

[57] And is the implementation used in openSMILE.

[58] And also implemented in the `cFunctionalMeans` component.

[59] In openSMILE the `norm` option of `cFunctionalMeans` can be set to *segment* to normalise counts and times etc. by N.

[60] Implemented in openSMILE in the `cFunctionalMoments` component.

The second order statistical moment, also called *variance* σ^2, is defined as:

$$m_2 = \sigma^2 = \frac{1}{N}\sum_{n=0}^{N-1}\left(x(n) - \mu_a\right)^2 . \qquad (2.260)$$

The standard deviation σ is defined as the root of the variance:

$$\sigma = \sqrt{\frac{1}{N}\sum_{n=0}^{N-1}\left(x(n) - \mu_a\right)^2}. \qquad (2.261)$$

A variation of the standard deviation which neutralises the influence of the mean, is the Coefficient of Variation (CV) (cf. Reed et al. 2002, for example), or normalised standard deviation:

$$\overline{\sigma} = \frac{\sigma}{\mu_a}. \qquad (2.262)$$

Using the CV is only recommended for variables with an expected mean way greater than zero (this is not true, however, if σ is also very small). It is undefined for $\mu_a = 0$. To avoid invalid values in large-scale data-analysis in this thesis, in the case of $\mu_a = 0$ the following substitution is used: $\overline{\sigma} = \sigma$.

Higher order statistical moments are of less relevance, but for the completeness of the presented set of features, order three and four are considered nonetheless. The third central moment m_3 is defined as:

$$m_3 = \frac{1}{N}\sum_{n=0}^{N-1}\left(x(n) - \mu_a\right)^3 . \qquad (2.263)$$

Due to the third power in the sum, the values of m_3 can be very large (or close to zero). Thus, instead of the central moment m_3, the third standardised moment $\overline{m}_3$ is used here (commonly known as *skewness*):

$$\overline{m}_3 = \frac{m_3}{\sigma^3}. \qquad (2.264)$$

In the same way the fourth central and fourth standardised moment are defined:

$$m_4 = \frac{1}{N}\sum_{n=0}^{N-1}\left(x(n) - \mu_a\right)^4 , \qquad (2.265)$$

$$\overline{m}_4 = \frac{m_4}{\sigma^4}. \qquad (2.266)$$

The fourth standardised moment is also known as *kurtosis*.

2.4.2.3 Extreme Values

Extreme values[61] can be important markers in a signal, i.e., the maximum pitch value or the maximum energy value. In this thesis the following functionals related to global extrema have been considered: global maximum and minimum value of $x(n)$ (x_{max}, x_{min}), the positions (indices) of the global maximum and minimum value (n_{max}, n_{min}), and the range R_x of the signal: $R_x = x_{max} - x_{min}$.

Two additional functionals are formulated in this thesis, based on the maximum and minimum values: the difference $d_{max,\mu}$ between the maximum value and the arithmetic mean ($d_{max,\mu} = x_{max} - \mu_a$) and the difference $d_{min,\mu}$ between the arithmetic mean and the minimum value ($d_{\mu,\,min} = \mu_a - x_{min}$).

2.4.2.4 Percentiles

The arithmetic mean, and especially the global extreme values (maximum and minimum) are all sensitive to outliers, i.e., single values which are well out of the typical range of the majority of values. If there is one single very large value which has been caused by noise or corruption of the input, the maximum value (and its position) will relate to this value (erroneously). Instead, for more robustness against outliers, percentiles should be considered.[62] The jth percentile P_j is defined as the value x below which j percent of all the values $x(n)$ are, i.e., for j percent of the values in $x(n)$ the following is true: $x(n) < P_j$.

For computational efficiency when computing multiple percentiles of the same time series $x(n)$, the series is sorted in ascending order (denoted here as $s(n)$). The jth percentile can the be found in the sorted set of values at the index n_j, i.e., $P_j = s(n_j)$ with

$$n_j = \lfloor \frac{j}{100}(N-1) + 0.5 \rfloor. \tag{2.267}$$

This equation is inprecise if N is small and N is not evenly dividable by j, i.e., when the actual percentile value lies between $s(n_j)$ and either $s(n_j - 1)$ or $s(n_j + 1)$. For these cases a linear interpolation method to improve the precision of the percentile values has been implemented. With the the real valued (exact) index of the percentile P_j location n'_j, and the lower and upper bound integer indices ($n_{j,l}$ and $n_{j,u}$, respectively), the weights $w_{j,l}$ and $w_{j,u}$ for the values $s(n_{j,l})$ and $s(n_{j,u})$ can be computed:

$$n'_j = \frac{j}{100}(N-1), \tag{2.268}$$

$$n_{j,l} = \lfloor n'_j \rfloor, \tag{2.269}$$

[61] In openSMILE extreme values can be extracted with the `cFunctionalExtremes` component.

[62] Percentiles are implemented in openSMILE in the `cFunctionalPercentiles` component.

$$n_{j,u} = \lceil n'_j \rceil, \tag{2.270}$$

$$w_{j,l} = n'_j - n_{j,l}, \tag{2.271}$$

$$w_{j,u} = n_{j,u} - n_j. \tag{2.272}$$

With these weights, the linearly interpolated jth percentile value is expressed as:

$$P'_j = \begin{cases} w_{j,l}s(n_{j,l}) + w_{j,u}s(n_{j,u}) & n_{j,l} \overset{!}{=} n_{j,u} \\ s(n_{j,l}) & n_{j,l} = n_{j,u} \end{cases}. \tag{2.273}$$

The median of a set of values (e.g., a time series $x(n)$) is equivalent to the 50th percentile of that set of values. Other well known percentiles are the quartiles I–III, which correspond to the 25th, 50th, 75th percentile and the quintiles, which correspond to the 20th, 40th, 60th, and 80th percentile.

Further, percentile ranges have been often used in related work (Schuller 2006, 2013). The range between percentile j and percentile i is defined as the difference between the respective percentile values: $P^r_{j,i} = P_j - P_i$ for $j > i$. Special cases of percentile ranges are the three Inter-Quartile Ranges (IQRs): IQR1-2 is $P_{50} - P_{25}$, IQR2-3 is $P_{75} - P_{50}$ and IQR1-3 is $P_{75} - P_{25}$.

In order to robustly express maximum and minimum signal values for noisy signals, it is suggested here to use the 95th and 05th percentile instead of the maximum and minimum value, respectively. Alternatively, for longer segments ($\geq \approx 2$ s), the 99th and 01st percentile can be used.

2.4.2.5 Temporal Centroid

Similar to the spectral centroid (Sect. 2.2.4.6), the temporal centroid $t_{centroid}$ of the signal $x(n)$ can be computed as[63]:

$$x_{centroid} = \frac{\sum_{n=0}^{N-1} t(n)x(n)}{\sum_{n=0}^{N-1} x(n)}, \tag{2.274}$$

where $t(n)$ is a time-scaling function that can be used to modify the time units. For a time scale in seconds $t(n) = nT_f$, for a time-scale in frames $t(n) = n$, and for a time scale in relative percentage of the segment $t(n) = n/N$.

[63] In openSMILE the temporal centroid is implemented by the `cFunctionalRegression` component, as the sums are shared with the regression equations, thus computing both descriptors in the same component increases the efficiency.

2.4.2.6 Regression

In order to describe the general shape of the time series $x(n)$, linear and quadratic regression coefficients can be used.[64]

For the linear regression the values of the series $y = x(n)$ are approximated by a line $\hat{y} = mn + o$ in a way that the quadratic error between the line and the actual series is minimised. The regression coefficients m and o[65] are sometimes also known as m (slope) and t (offset), respectively, or plainly a and b, respectively. The derivation for the linear regression coefficients is the same as for the spectral slope (Sect. 2.2.4.2). It is thus, not repeated here. A generalisation for different time-scales (other than frame index n) can be easily obtained by substituting a general time-scale function $t(n)$ for the frame index n in all equations and $L' = t(N)$ for the length of the sequence $x(n)$ on the new time scale. The solutions for m and o for the general case of $y = x(t(n))$ are as follows:

$$m = \frac{N\Sigma_{ty} - \Sigma_t \Sigma_y}{N\Sigma_{t^2} - \Sigma_t^2}, \tag{2.275}$$

$$o = \frac{1}{N-1}\left(\Sigma_y - \frac{\Sigma_{ty}}{\Sigma_t}\right), \tag{2.276}$$

where

$$\Sigma_y = \sum_{n=0}^{N-1} x(n), \tag{2.277}$$

$$\Sigma_t = \sum_{n=0}^{N-1} t(n), \tag{2.278}$$

$$\Sigma_{t^2} = \sum_{n=0}^{N-1} t(n)^2, \tag{2.279}$$

$$\Sigma_{ty} = \sum_{n=0}^{N-1} t(n)x(n). \tag{2.280}$$

For a linear time scale $t(n) = gn$ with a general time-stretch factor g, the following simplifications to speed up the computations can be made (using exponential series expansions from Rade et al. 2000, p. 189):

$$\Sigma_t = g\frac{1}{2}N(N-1), \tag{2.281}$$

[64] In openSMILE the `cFunctionalRegression` component computes linear and quadratic regression coefficients.

[65] As used in this thesis, in order to avoid a name conflict with the quadratic regression coefficients a and b and time t.

$$\Sigma_{t^2} = g^2 \frac{1}{6} N(N-1)(2N-1). \tag{2.282}$$

The unit of the slope m is the unit of $x(n)$ (e.g., an amplitude, or power unit, or Hz for a frequency) per time-unit (unit of $t(n)$, e.g., frames, seconds, or relative segment percentage). The unit of the offset b is the same as the unit of $x(n)$.

For the quadratic regression the values of the series $y = x(t(n))$ are approximated by a parabola $\hat{y} = \hat{x}(t(n)) = at(n)^2 + bt(n) + c$. The derivation of the least-squares fit of the parabola to the series follows the same pattern as for the linear regression: The minimisation of the quadratic error e^2 between the quadratic approximation of the function and the function itself is expressed as:

$$\begin{aligned} e^2 &= \sum_{n=0}^{N-1} \big(x\left(t\left(n\right)\right) - \hat{x}\left(t\left(n\right)\right)\big)^2 = \\ &= \sum_{n=0}^{N-1} \big(x\left(t\left(n\right)\right) - at\left(n\right)^2 - bt\left(n\right) - c\big)^2 \overset{!}{=} min. \end{aligned} \tag{2.283}$$

From this, the following three differential equations for a, b, and c for the points $\big(t\left(n\right), x\left(t\left(n\right)\right)\big)$ with $n = 0 \ldots N-1$ are obtained:

$$\frac{\delta}{\delta a} e^2 = \sum_{i=0}^{N-1} -2t(n)^2 \big(x(t(n)) - at(n)^2 - bt(n) - c\big) \overset{!}{=} 0, \tag{2.284}$$

$$\frac{\delta}{\delta b} e^2 = \sum_{i=0}^{N-1} -2t(n) \big(x(t(n)) - at(n)^2 - bt(n) - c\big) \overset{!}{=} 0, \tag{2.285}$$

$$\frac{\delta}{\delta c} e^2 = \sum_{i=0}^{N-1} -2 \big(x(t(n)) - at(n)^2 - bt(n) - c\big) \overset{!}{=} 0. \tag{2.286}$$

Solving these three equations for a, b, and c yields solutions for these parameters in terms of the given points $\big(t(n), x(t(n))\big)$. With

$$\Sigma_{t^2 y} = \sum_{n=0}^{N-1} t(n)^2 y(n), \tag{2.287}$$

$$\Sigma_{t^3} = \sum_{n=0}^{N-1} t(n)^3, \tag{2.288}$$

$$\Sigma_{t^4} = \sum_{n=0}^{N-1} t(n)^4, \tag{2.289}$$

the above system of equations can be re-written in matrix form:

$$\begin{pmatrix} \Sigma_{t^4} & \Sigma_{t^3} & \Sigma_{t^2} \\ \Sigma_{t^3} & \Sigma_{t^2} & \Sigma_t \\ \Sigma_{t^2} & \Sigma_t & N \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} \Sigma_{t^2y} \\ \Sigma_{ty} \\ \Sigma_y \end{pmatrix}. \tag{2.290}$$

Again, for a linear time scale $t(n) = gn$, the following substitutions can be applied to speed up the computation (using exponential series expansions from Rade et al. 2000, p. 189):

$$\Sigma_{t^3} = \frac{g^3}{4} N^2 (N-1)^2, \tag{2.291}$$

$$\Sigma_{t^4} = \frac{g^4}{5}(N-1)^5 + \frac{1}{4}(N-1)^4 + \frac{1}{3}(N-1)^3 - \frac{1}{30}(N-1). \tag{2.292}$$

The solution is then given by Cramer's rule (Rade et al. 2000, p. 93) as:

$$a = \frac{1}{d}\begin{vmatrix} \Sigma_{t^2y} & \Sigma_{t^3} & \Sigma_{t^2} \\ \Sigma_{ty} & \Sigma_{t^2} & \Sigma_t \\ \Sigma_y & \Sigma_t & N \end{vmatrix}, \tag{2.293}$$

$$b = \frac{1}{d}\begin{vmatrix} \Sigma_{t^4} & \Sigma_{t^2y} & \Sigma_{t^2} \\ \Sigma_{t^3} & \Sigma_{ty} & \Sigma_t \\ \Sigma_{t^2} & \Sigma_y & N \end{vmatrix}, \tag{2.294}$$

$$c = \frac{1}{d}\begin{vmatrix} \Sigma_{t^4} & \Sigma_{t^3} & \Sigma_{t^2y} \\ \Sigma_{t^3} & \Sigma_{t^2} & \Sigma_{ty} \\ \Sigma_{t^2} & \Sigma_t & \Sigma_y \end{vmatrix}, \tag{2.295}$$

$$d = \begin{vmatrix} \Sigma_{t^4} & \Sigma_{t^3} & \Sigma_{t^2} \\ \Sigma_{t^3} & \Sigma_{t^2} & \Sigma_t \\ \Sigma_{t^2} & \Sigma_t & N \end{vmatrix}. \tag{2.296}$$

Regression coefficients obtained with the above equations, where $t(n) = n$ (frame index time scale) can be converted to another linear time-scale $t(n) = gn$ with the linear scale factor g by the following transformations:

$$m' = g^{-1}m, \tag{2.297}$$

$$a' = g^{-2}a, \tag{2.298}$$

$$b' = g^{-1}b. \tag{2.299}$$

The scale factor g can be set to $g = T_f$ for an absolute time scale in seconds, or to $g = 1/N$ for a percentage time scale relative to the segment length N.[66] The derivation

[66] In openSMILE, the time scaling feature is enabled by the `normRegCoeff` option in `cFunctionalRegression` component. Setting it to 1 enables the relative time scale $g = 1/N$ and setting it to 2 enables the absolute time scale in seconds.

of these transformations is obtained by setting $t(n) = gn$ in Eqs. (2.278)–(2.280) and in Eqs. (2.287)–(2.289) and re-evaluating Eqs. (2.275)–(2.276) and Eqs. (2.293)–(2.295). In this way it is also shown that linear scaling of the time scale has no influence on the linear regression coefficients o (linear offset), and the quadratic regression offset c.

If the coefficients must be independent of the range R_x of values in $x(n)$, the coefficients m, a, and b must be divided by R_x:

$$m^{(norm)} = \frac{m}{R_x}. \tag{2.300}$$

Equation (2.300) also applies to a and b. The offset coefficients o and c must be transformed according to (also applies for c)[67]:

$$o^{(norm)} = \frac{o - x_{min}}{R_x}. \tag{2.301}$$

The derivations are made by setting

$$x^{(norm)}(t(n)) = \frac{1}{R_x}\left(x(t(n)) - x_{min}\right) \tag{2.302}$$

in the regression coefficient equations.

Besides the linear or quadratic regression coefficients, an informative measure is the regression error e, i.e., the remaining (minimised) difference between the computed regression line or curve $\hat{x}(t(n))$ and the actual time series $x(t(n))$ (Eq. (2.283)). The error is normalised by the length of the series to obtain a measure $\bar{e}$ which is independent of the sequence length:

$$\bar{e} = \frac{1}{N}e. \tag{2.303}$$

The error is either called *linear regression error* when it is computed as difference between $x(n)$ and a line, or *quadratic regression error*, when it is a difference between $x(n)$ and a parabola. As alternative to the normalised sum-square error $\bar{e}$ (Eq. (2.283)), the normalised absolute error $\bar{e}_a$ is additionally used:

$$\bar{e}_a = \frac{1}{N}\sum_{n=0}^{N-1}\left|x(t(n)) - \hat{x}(t(n))\right|. \tag{2.304}$$

[67] Option `normInputs` in openSMILE component `cFunctionalRegression`—also affects linear and quadratic error.

The regression errors can be normalised to the range R of $x(n)$ in order to have comparable errors across segments with different magnitudes[68]:

$$\overline{e}_a^{(norm)} = \frac{1}{R}\overline{e}_a, \tag{2.305}$$

$$\overline{e}^{(norm)} = \frac{1}{R^2}\overline{e}. \tag{2.306}$$

For quadratic regression, also the (relative) temporal location n_v of the parabola vertex and its corresponding amplitude $\hat{x}(n_v)$ are of interest. If the vertex is within $n = 0 \dots N-1$, then also the slope of the line connecting the point at $n = 0$ with the vertex (left slope) and the slope of the line connecting the vertex with the last point at $n = N-1$ (right slope) can be computed. The vertex coordinates n_v and $\hat{x}(n_v)$ are computed according to the following equations (cf. Sect. 2.2.11.4—quadratic interpolation):

$$n_v = \frac{b}{-2a}, \tag{2.307}$$

$$\hat{x}(n_v) = c - \frac{b^2}{4a}. \tag{2.308}$$

To avoid outliers, the range of n_v is limited to $[0; N-1]$, thus the corresponding $\hat{x}(n_v)$ is actually computed as:

$$\hat{x}(n_v) = an_v^2 + bn_v + c. \tag{2.309}$$

The parabola vertex coordinates can be normalised to the scale $n = 0 \dots 1$ (for n_v) and to the range R of the values in $x(n)$ (for $\hat{x}(n_v)$) by:

$$n_v^{(norm)} = \frac{n_v}{N-1}, \tag{2.310}$$

$$\hat{x}(n_v)^{(norm)} = \frac{\hat{x}(n_v) - min}{R}. \tag{2.311}$$

For the computation of linearised left and right slopes of the parabola, the parabola is evaluated for $n = 0$, yielding $\hat{x}(0) = c$, and for $N-1$, yielding $\hat{x}(N-1) = a(N-1)^2 + b(N-1) + c$. The left slope m_{left} is then formulated as:

$$m_{left} = \begin{cases} \dfrac{\hat{x}(n_v) - c}{n_v} & n_v > 0 \\ 0 & n_v \leq 0 \end{cases}, \tag{2.312}$$

and the right slope m_{right} is given as:

[68] Option `normInputs` in the openSMILE component `cFunctionalRegression`—note that this option also affects the regression coefficients as it effectively normalises the input range.

$$m_{right} = \begin{cases} \dfrac{\hat{x}(N-1) - \hat{x}(n_v)}{N-1-n_v} & n_v < N-1 \\ 0 & n_v \geq N-1 \end{cases}. \qquad (2.313)$$

The slopes normalised to the value range, or the scale $n = 0 \ldots 1$, or both, are given by using the respective normalised values in Eqs. (2.312) and (2.313):

$$m_{left}^{(norm)} = \begin{cases} \dfrac{\hat{x}(n_v)^{(norm)} - c^{(norm)}}{n_v^{(norm)}} & n_v > 0 \\ 0 & n_v \leq 0 \end{cases}, \qquad (2.314)$$

$$m_{right}^{(norm)} = \begin{cases} \dfrac{\hat{x}(N-1)^{(norm)} - \hat{x}(n_v)^{(norm)}}{1 - n_v^{(norm)}} & n_v < 1 \\ 0 & n_v \geq 1 \end{cases}. \qquad (2.315)$$

An important property of the regression coefficient functionals is the *irreversibility*, i.e., that if the input sequence is reversed, a different value is obtained (except for the rare case of exactly symmetric input series and quadratic regression coefficients). Most other functionals, such as the means, moments, extreme values (excluding positions), temporal distributions, peak amplitudes, and some of the modulation features, will return the same value even if the input sequence is reversed in time.

2.4.2.7 Temporal Distribution

The temporal distribution of the signal is described by statistics which give the percentage of values $x(n)$ in the time series which are above or below a certain threshold x_θ. Hence, these functionals are assigned the name *up-/down-level times*, respectively.[69]

In this thesis a range-relative threshold x_θ^{rel} is used to ensure independence from variations of the signal dynamics. x_θ^{rel} can be in the range [0; 1]. The absolute threshold x_θ is computed dynamically from the range (R_x) and minimum value (x_{min}) of $x(n)$ and the relative threshold:

$$x_\theta = x_{min} + x_\theta^{rel} R_x. \qquad (2.316)$$

Two kinds of threshold-based temporal descriptors are implemented: the *up-level* and *down-level times*. The *up-level time* is defined as the time or number of frames (absolute or relative)[70] the signal is above the threshold ($x(n) > x_\theta$). The *down-level time* is defined as the time or number of frames (again, absolute or relative) the signal is below or equal to the threshold ($x(n) \leq x_\theta$). Typical relative thresholds which are considered in standard feature sets are 0.25, 0.5, 0.75, and 0.9. The corresponding

[69] In openSMILE these functionals are implemented in the component `cFunctionalTimes`.

[70] Configurable with the `norm` option in openSMILE.

descriptors are then referred to as the 25, 50, 75, and 90 % up- or down-level times, respectively. The up- and down-level times with the same threshold are always complementary, i.e., their sum is 1 (for relative times) or the length of $x(n)$ for absolute times. Due to this redundancy, it is sufficient to use one or the other (up or down) in a feature set.

Additionally, *rise time* and *fall time* are considered. These are defined as the time or number of frames (absolute or relative) the signal is rising or falling, i.e., $x(n-1) < x(n)$ or $x(n-1) > x(n)$, respectively. Similarly, the *left-curvature time* and *right-curvature time* are defined as the time or number of frames the signal has a left or right curvature, i.e., $x(n) - x(n-1) < x(n+1) - x(n)$ (left) or $x(n) - x(n-1) > x(n+1) - x(n)$, respectively. In contrast to the up- and down-level times the rise-/fall times and the curvature times are not complementary because the case of a flat signal ($x(n-1) = x(n)$ and $x(n+1) = x(n)$) is not considered in the definition, i.e., for a constant signal all four of these descriptors will be 0, while for a signal which represents a line, the curvature times will be 0.

2.4.2.8 Peaks and Valleys

Peaks (maximum values) and valleys (minimum values) can be important markers in a signal. Both the amplitude and the position of the peaks are valuable. Various peak based descriptors have been implemented for this thesis.[71] These descriptors are described in the following, assuming a given set of I peaks $(x_{p,i}; n_{p,i})$ with $i = 0 \ldots I-1$ and J valleys $(x_{v,j}; n_{v,j})$ with $j = 0 \ldots J-1$:

- Number of peaks (I)—optionally normalised to peaks per second,
- Arithmetic mean of the peak amplitudes $x_{\mu}^{(peaks)}$:

$$x_{\mu}^{(peaks)} = \frac{1}{I} \sum_{i=0}^{I-1} x_{p,i}, \tag{2.317}$$

- Absolute peak amplitude range (maximum peak amplitude - minimum peak amplitude),
- Peak amplitude range normalised to the input contour's arithmetic mean,
- Difference of the arithmetic mean of the peak amplitudes to the arithmetic mean of $x(n)$,
- Ratio of the arithmetic mean of the peak amplitudes to the arithmetic mean of $x(n)$ (relative peak mean),
- Mean of peak to peak amplitude differences,
- Mean of peak to peak amplitude differences normalised by range R_x,
- Standard deviation of peak to peak amplitude differences,
- Standard deviation of peak to peak amplitude differences normalised by range R_x,

[71] In openSMILE these functionals can be applied with the `cFunctionalPeaks2` component; the `cFunctionalPeaks` component contains an older, obsolete peak picking algorithm.

- Mean distance between peaks $d_{\mu}^{(peaks)}$:

$$d_{\mu}^{(peaks)} = \frac{1}{I}\sum_{i=1}^{I-1}\left(n_{p,i} - n_{p,i-1}\right), \tag{2.318}$$

- Mean of difference of consecutive peak distances,
- Standard deviation of inter-peak distances $d_{\sigma}^{(peaks)}$:

$$d_{\sigma}^{(peaks)} = \frac{1}{I}\sum_{i=1}^{I-1}\left(n_{p,i} - n_{p,i-1} - x_{\mu}^{(peaks)}\right)^2, \tag{2.319}$$

- Arithmetic mean of the valley amplitudes $x_{\mu}^{(minima)}$:

$$x_{\mu}^{(minima)} = \frac{1}{J}\sum_{j=0}^{J-1} x_{m,j}, \tag{2.320}$$

- Absolute valley amplitude range,
- Valley amplitude range normalised to the input contour's arithmetic mean,
- Difference of the arithmetic mean of $x(n)$ to the arithmetic mean of the valley amplitudes,
- Ratio of the arithmetic mean of the valley amplitudes to the arithmetic mean of $x(n)$,
- Mean of valley to valley amplitude differences
- Mean of valley to valley amplitude differences normalised by range R_x,
- Standard deviation of valley to valley amplitude differences,
- Standard deviation of valley to valley amplitude differences normalised range R_x,
- Arithmetic mean of rising slopes, i.e., the slopes of the lines connecting a valley with the following peak. The rising slope $m_{j,i}$ for valley j and peak i, where $i = j$ (if $x(n)$ starts with a valley) or $i = j + 1$ (if $x(n)$ starts with a peak), is given by:

$$m_{j,i} = \frac{x_{p,i} - x_{v,j}}{T\left(n_{p,i} - n_{v_i}\right)}, \tag{2.321}$$

where T is a normalisation factor. T is equivalent to the frame period length (in seconds) for all experiments in this thesis[72] which results in a unit for the slope as *amplitude difference per second*. T can, however, also be set to $T = 1$ to obtain a slope in terms of amplitude difference per frame[73] or $T = 1/N$ for a slope in terms of amplitude difference per relative segment length.[74] The first ($x(0)$) and

[72] In openSMILE in `cFunctionalPeaks2 norm=second` has to be set for this behaviour (default).

[73] `norm=frame` in openSMILE.

[74] `norm=segment` in openSMILE.

last ($x(N-1)$) value of the signal x are treated as first/last valley/peak (the latter depending on whether the respective value is followed/preceded by a peak/valley, respectively) in order to compute at least one slope, even if 0 or only 1 peak (or valley) is present in the signal.
- Standard deviation of rising slopes,
- Maximum rising slope,
- Minimum rising slope,
- Arithmetic mean of (positive) falling slopes, i.e., the slopes of the lines which connect a peak with the following valley. The positive falling slope $m_{i,j}$ for peak i and valley j, where $j = i$ (if $x(n)$ starts with a peak) or $j = i + 1$ (if $x(n)$ starts with a valley), is given by:

$$m_{i,j} = \frac{x_{p,i} - x_{v,j}}{T\left(n_{v,j} - n_{p_i}\right)\cdot}, \tag{2.322}$$

  where T is a time-domain normalisation factor (see *rising slope* above for details),
- Standard deviation of falling slopes,
- Maximum falling slope,
- Minimum falling slope.

The inter peak and valley distances measured in frames, and their standard deviations, can be converted to either distances in seconds by multiplying with the frame period length T (in seconds), or to distances measured as proportions of the segment length N by dividing by N.[75] Normalising parameters to a time scale of seconds is recommended for all peak and valley related functionals because this makes the descriptors independent of the input length and the underlying frame rate.

To find the peaks, a peak picking algorithm must be applied. The most simple algorithm is the extreme value search for maxima ($x(n-1) < x(n) > x(n+1)$), which finds all local maximum values. However, in speech and music signals small local extreme values might occur due to windowing artefacts or noise. These values are insignificant and have no global meaning. Thus, a threshold should be used to discard extreme values with a low local/relative amplitude. The algorithm implemented for this thesis consists of three steps. It

1. finds all local maxima and minima (extrema) in $x(n)$,
2. discards extrema with a low relative amplitude (based on an absolute threshold which is a configurable fraction of the range of $x(n)$—see Eq. (2.316)),[76] and
3. enforces a constraint of alternating maxima/minima, i.e., it discards two adjacent maxima with no valid (!) minimum in between and vice versa.

[75] In openSMILE the `norm` option controls this behaviour (frames, seconds, segment—respectively).

[76] See the `absThresh` and `relThresh` options in the openSMILE component `cFunctionalPeaks2`.

2.4.2.9 Segments

Some signals can be divided into meaningful, continuous segments, such as voiced and unvoiced parts of the pitch contour, or high and low signal energy regions. The shape and structure of these segments contains valuable paralinguistic information, in particular temporal and rhythmic information. Based on unsupervised statistical segmentation algorithms, segments based on statistics can be found in any arbitrary signal (e.g., MFCC, etc.).[77] Depending on the segmentation algorithm and the meaning of the segments when considering the underlying LLD, segments are either defined as adjacent, i.e., the end of the first segment is identical to the beginning of the second segment, or as separated by gaps, i.e., between segment one and two there is a gap of at least one sample/frame length. The following segmentation algorithms which return adjacent segments are considered in this thesis:

delta threshold A segment boundary is placed wherever $x(n) > \overline{x}(n) + \delta_x$, with $\overline{x}(n)$ being the sliding window moving average over w frames (Schuller 2006). w is estimated from the maximum number of allowed segments $N_{seg,\,max}$ as $w = 2N/N_{seg,\,max}$. If required, w can also be set to a fixed custom value[78]—however, this has not been applied in the experiments in this thesis. δ_x is the threshold that has to be exceeded in order for a new segment to begin. Is is computed from a range-relative threshold θ_R as $\delta_x = R_x\theta_R$.

relative threshold A segment boundary is placed wherever $x(n)$ or a running average $\overline{x}$ (sliding window, size $w = 3$ frames) of $x(n)$ crosses or touches a pre-defined threshold a, i.e., the boundary is placed at frame n, when $\overline{x}(n-1) \leq a < \overline{x}(n)$ or $\overline{x}(n-1) \geq a > \overline{x}(n)$. The absolute threshold a can be computed from a range-relative threshold θ_R:

$$a = x_{min} + R_x\theta_R, \tag{2.323}$$

where R_x is the range of $x(n)$, or from a mean-relative threshold θ_μ:

$$a = \mu_x\theta_\mu, \tag{2.324}$$

where μ_x is the arithmetic mean of $x(n)$.

change from/to constant A segment boundary is placed wherever $x(n)$ changes from $x(n) = a$ to $x(n) \neq a$ or vice versa.

Further, the following algorithms which return segments of interest which are separated by gap segments are implemented:

unequal to Segments are separated by continuous regions in which $x(n) = a$. Within a segment, $x(n) \neq a$ must be true for the samples within the segment (except for small gaps, see below). a is a configurable threshold parameter and depends on the type of input (LLD) the functional is applied to. In this thesis, this

[77] In openSMILE segment-based temporal functionals can be computed with the component `cFunctionalSegments`.

[78] Use the `ravgLng` option of the `cFunctionalSegments` component in openSMILE.

functional is applied to the F_0 LLD, for example, and $a = 0$. Thus, the segments considered correspond to voiced segments where by definition $F_0 > 0$ is true. In order to increase noise robustness (e.g., robustness to F_0 extraction errors), a minimum gap (pause) duration of 2 frames is enforced.[79] If a gap is shorter or equal to the minimum pause duration, the segments left and right of the short gap are considered as a single segment, including the gap.

equal to In analogy to the *unequal to* condition, segments are separated by continuous regions where $x(n) \neq a$, and $x(n) = a$ is true within a segment.

Additionally, for all algorithms, a minimum segment length constraint is enforced to avoid segments of one frame length due to noise. The minimum segment length $l_{seg,\,min}$ is estimated from the maximum number of allowed segments $N_{seg,\,max}$: $l_{seg,\,min} = N/N_{seg,\,max} - 1$.

From the segment boundaries, the following temporal descriptors are computed:

- Number of segments,
- Arithmetic mean of segment lengths,
- Standard deviation of segment lengths,
- Maximum segment length,
- Minimum segment length,

If the segments are not defined as adjacent segments, but as segments separated by gaps, the following functionals can be additionally computed:

- Arithmetic mean of the gaps between segments,
- Standard deviation of the gaps between segments,
- Maximum length of gap between segments,
- Minimum length of gap between segments.

2.4.2.10 Onsets

An onset is generally defined as a sudden, steep amplitude rise in $x(n)$, e.g., the beginning/onset of an acoustic event. Here, onsets as a functional follow a more simple definition, similar to the threshold definition for segment based functionals (Sect. 2.4.2.9): An onset is marked at each position n where $x(n)$ rises above a given absolute threshold a: $x(n-1) \leq a < x(n)$. An "offset" marks the inverse, i.e., the case when $x(n)$ falls below a: $x(n-1) > a \geq x(n)$. For some LLDs it might make sense to use the absolute value of $x(n)$ to define onsets only by the magnitude of the descriptor. While this has not been used in this thesis, as onset functionals have been applied to F_0 contours only, the equations for the conditions are given here as: $|x(n-1)| \leq a < |x(n)|$ for an onset and $|x(n-1)| > a \geq |x(n)|$ for an "offset".

[79] This length can be changed via the `pauseMinLng` option of the `cFunctionalSegments` component.

The following functionals are computed from the onset and "offset" positions[80]:

- The position of the first onset, either as frame number n, relative to the segment length $n_{rel} = n/N$, or as a time in seconds $t = n \cdot T$,
- The position of the last "offset", either as frame number n, relative to the segment length $n_{rel} = n/N$, or as a time in seconds $t = n \cdot T$,
- The absolute number of onsets N_{onsets},
- The absolute number of "offsets" $N_{offsets}$ (only provided for completeness, but not used, as it is highly redundant with the number of onsets—a more informative feature for future work would be the difference between onsets and "offsets", which can be either -1, 0, or $+1$, depending on whether $x(n)$ starts/ends with a value above/below the threshold a),
- The onset rate (frequency) in Hz: $f_{onsets} = \frac{N_{onsets}}{N \cdot T}$.

2.4.2.11 Crossings

With the definition of ZCR from Sect. 2.2.1.1, ZCR can be applied as a functional.[81] Since—in contrast to time domain audio signals—most LLDs are not symmetric around zero, the MCR is favoured. The MCR is defined like the ZCR but computed from $x(n)$ after mean removal as given by Eq. (2.37).

2.4.2.12 Sample Values

In order to capture sample values of $x(n)$ at given temporal positions independent of the length of $x(n)$, values are sampled at given relative time instants n_{rel}, where n_{rel} is in the range [0; 1] and gives the sample time as percentage of the length of $x(n)$. Thereby the absolute position n is computed as $n = N \cdot n_{rel}$. This functional is recommended for rather short segments, such as isolated phonemes, words, beats or bars. In this case the relative positions have meanings, such as the beginning, middle, and end of the segment (for 3 sample values placed accordingly).[82]

2.4.3 Modulation Functionals

Modulation features have recently gained attention in speech related tasks (cf. Mubarak et al. 2006; Wu et al. 2011; Chi et al. 2012) and have been inherently used for rhythmic analysis in Music Information Retrieval (Mubarak et al. 2006; Schuller et al. 2007).

[80] Computed in openSMILE by the `cFunctionalOnset` component.

[81] Provided by the `cFunctionalCrossings` component in openSMILE.

[82] Sample-based functionals are provided by the `cFunctionalSamples` component in openSMILE.

The goal of modulation features is to qualitatively describe the modulations and periodicities of a signal $x(n)$, as well as the shapes thereof with a set of numerical descriptors. For this thesis the following types of modulation functionals applicable to LLD signals have been considered: Discrete Cosine Transformation coefficients (Sect. 2.4.3.1), LPC coefficients (Sect. 2.4.3.2), and modulation spectrum coefficients (Sect. 2.4.3.3).

2.4.3.1 Discrete Cosine Transformation Coefficients

For the DCT coefficient functional,[83] the DCT base functions from order o_0 to order o_l are computed and applied to the signal to obtain the $o_l - o_0 + 1$ DCT coefficients from order o_0 to o_l. The base functions are expressed relative to the length N of the input $x(n)$. This leads to functionals which describe the overall contour of each input segment as if all segments were normalised to a common length. Thus, the DCT coefficients do not correspond to absolute frequencies, but rather frequencies relative to the segment length. For modulation coefficients corresponding to absolute frequencies, independent of the segment length, the modulation spectrum coefficients (Sect. 2.4.3.3) were designed.

The same DCT-II as used in Sect. 2.2.10.1 for MFCC is applied here. The oth DCT base function $b_{DCT}^{(o)}(n)$ is thus given as $(n = 0 \ldots N-1)$:

$$b_{DCT}^{(o)}(n) = \cos\left(\pi \frac{n}{N}(o + 0.5)\right). \tag{2.325}$$

The oth DCT coefficient is then computed as:

$$DCT(o) = \sqrt{\frac{2}{N}} \sum_{n=0}^{N-1} x(n) b_{DCT}^{(o)}(n). \tag{2.326}$$

Typically DCT coefficients 1–5 are used as functionals because most affective and paralinguistic information is hidden in the lower frequency modulations, which correspond to the lower order DCT coefficients. If the length of segments varies strongly, DCT coefficient functionals should not be used, due to reasons discussed above.

2.4.3.2 Linear Predictor Coefficients

Although Linear Predictive Coding has been derived from the source-filter model of speech production, linear prediction is a general method to model linear dependencies present among subsequent samples of an arbitrary signal $x(n)$. LP analysis can therefore be used as a functional on LLD signals.[84]

[83] In openSMILE the `cFunctionalDCT` component computes DCT coefficient functionals.

[84] In openSMILE the `cFunctionalLpc` component computes LP-analysis functionals.

LPC coefficients are computed from a signal $x(n)$ (length N) with the autocorrelation method which is described in Sect. 2.2.7.1. In most experiments reported on in this thesis the coefficients a_i with $i = 1 \ldots 6$ are used, as well as the gain of LP analysis (the energy of the remaining error signal). A low gain indicates a signal which can be well approximated by LP analysis, i.e., a deterministic signal, while a high gain indicates a stochastic signal.

As most LLDs are sampled at a constant rate of 100 Hz, LP coefficients are independent of the segment length and can be used when the segment length varies strongly—in contrast to the DCT coefficients. I.e., if a signal of $N/2$ frames is appended to itself to obtain a sequence of length N, the resulting LP coefficient functionals will be very similar to those LP coefficients computed from only the $N/2$ length signal. The resulting DCT coefficients, however, will differ significantly across both cases. More precisely, they will be shifted by one order, i.e., $DCT(1)$ of the $N/2$ length signal will be roughly equal to $DCT(2)$ of the full (N) length signal.

2.4.3.3 Modulation Spectrum

The modulation spectrum[85] of a (LLD) signal $x(n)$ of length N is computed via a Short-Time Fourier Transform (STFT) approach (short-time analysis, cf. Sect. 2.1.3). Thereby the signal $x(n)$ is divided into I frames (each N_f samples long), a windowing function (cf. Sect. 2.1.3) is applied to the frames, zero-padding is performed, and a FFT is applied to each windowed signal $x_i(n)$ ($i = 1 \ldots I$) and the linear magnitudes of the resulting complex spectrum are computed (cf. Sect. 2.2.3.1). The magnitude spectra X_i of each short-time frame are averaged over all I frames, resulting in the raw modulation spectrum $X^{(mod,raw)}$. With a constant window size N_f, this method enables the efficient computation of the modulation spectrum on arbitrary length signals in linear time with respect to the signal length N.

As a last step, the spectrum $X^{(mod,raw)}$ is scaled and interpolated to a fixed number of bins M', a number which is independent of the window size N_f. This way it is ensured that modulation spectrum features computed with arbitrary (N_f, N) combinations are compatible. Spline interpolation (Steffensen 2012) is used to map $X^{(mod,raw)}(m)$ to $X^{(mod)}(m')$ ($m' = 1 \ldots M'$). The spline function is constructed with the points $(F(m), X^{(mod)}(m))$, where $F(m)$ maps a bin index m to a linear frequency (Hz) f (cf. Sect. 2.2.3.1). The spline function is then evaluated for the frequencies $f' = F'(m')$ corresponding to the bins of the spectrum $X^{(mod)}$ to obtain the respective (interpolated) magnitudes.

Phase information of the modulation spectrum is not considered here. In future work, however, this could be addressed when segments with meaningful start and end points are analysed, i.e., the segments are not selected as fixed length segments at a fixed rate from a stream. Such segments could be for speech: whole words, phrases,

[85] In openSMILE the `cFunctionalModulation` component computes modulation spectrum functionals.

parts of sentences, or full sentences, for example, and for music: beats, one or more bars, or the chorus, for example.

As a default recommendation, the frequency resolution δ_f for $X^{(mod)}$ was chosen between 0.1 to 0.5 Hz and a range from 0.25 to 30 Hz was considered by the author of this thesis, which results in $N_f = 298$ bins ($\delta_f = 0.1$) and $N_f = 60$ bins ($\delta_f = 0.5$).

The modulation spectrum can either be used as a feature vector directly, or further statistics can be computed from this spectrum. All statistics applicable to normal magnitude and power spectra can be applied. Most important, however, seem to be the frequency(-ies) related to the peak(s) of the modulation spectrum and the flatness of the modulation spectrum.[86]

2.4.3.4 Rhythmic Features

For music analysis, rhythmic features are of high relevance. While for speech, modulation spectrum, and LPC based modulation features might be sufficient, the highly structured rhythm of music requires features developed specifically for music rhythm analysis. Such features have been proposed by the author of this thesis in (Schuller et al. 2007), originally.[87] The features are derived by a 2-step comb-filterbank analysis of Mel-scale auditory band spectra. A so-called *Tatum vector* (resembling a modulation spectrum in a range which covers very fast tempi) and a *meter vector*, which is a modulation spectrum computed at multiples of the Tatum tempo, are computed for a music segment or a whole piece of music. The Tatum tempo is computed by finding the most prominent peak in the Tatum vector. For details, the reader is referred to Schuller et al. (2007) and Eyben and Schuller (2010b). By definition in (Schuller et al. 2007), the Tatum vector has 57 elements, and the meter vector has 19 elements, which covers 19 metrical levels, i.e., 19 multiples of the Tatum tempo.

2.5 Modelling

In order to evaluate the acoustic features and especially the standard feature sets introduced in the previous chapters, classification experiments were performed. This section gives a brief overview over the classification and modelling methods used in this thesis and a theoretical introduction to the two categories of modelling: static (Sect. 2.5.1) and dynamic (Sect. 2.5.2).

[86] In openSMILE, the statistics can be applied to the modulation spectrum with the `cSpectral` component. Also other components which expect magnitude spectra (e.g., ACF in `cAcf`) can read from the output of `cFunctionalModulation`.

[87] These features are not part of openSMILE (yet). It is planned to include them in future releases. C code is available from the author of this thesis upon request.

In the field of affect recognition and Computational Paralinguistics, most methods in the past have dealt with classification on an utterance level, e.g., Ververidis and Kotropoulos (2006); Vlasenko et al. (2007); Schuller et al. (2009b, 2013b). Thereby each utterance or a part of an utterance is associated with one set of affective labels and the classifier/regressor assigns exactly one set of predictions to one utterance. As shown by Vlasenko et al. (2007), for example, either a static classifier, such as a Support Vector Machine (SVM) (e.g., Schuller et al. 2006) or a dynamic classifier, such as a Hidden Markov Models (HMMs) can be used (e.g., Schuller et al. (2003)). The SVM estimates a classification label from a single high dimensional feature vector, which summarises the utterance. The HMM computes a likelihood score for every frame of low-level features and from the most likely path over the utterance estimates the most likely class. Recently, databases with dimensional affect ratings have emerged: the Sensitive Artificial Listener (SAL) set in the HUMAINE database (Douglas-Cowie et al. 2007), the SEMAINE database (Schröder et al. 2012), the RECOLA database (Ringeval et al. 2013), and a set of continuous music mood annotations (Soleymani et al. 2013). Tools like Feeltrace (Cowie et al. 2000) have been used for continuous rating, both in time and value. Such databases have caused a shift in methods, first of all moving from classification to regression to be able to model continuous affective dimensions (Grimm et al. 2007; Wöllmer et al. 2008), and next moving from utterance or segment level labels to quasi time-continuous labels (Eyben et al. 2010c, 2012; Schröder et al. 2012; Schuller et al. 2012b; Weninger et al. 2013, 2014)—creating a need for research on dynamic, context aware modelling methods. This thesis proposes and evaluates such a modelling method based on combining supra-segmental features with dynamic modelling by Long Short-Term Memory Recurrent Neural Networks (LSTM-RNNs) in Sect. 6.4.

In the field of MIR, static modelling has also been applied for several tasks such as dance-style recognition (Schuller et al. 2007), genre identification (e.g., Tzanetakis and Cook 2002; Eyben and Schuller 2010a), and chord and key transcription (e.g., Lee and Slaney 2008; Schuller et al. 2008). Music fingerprinting and identification of artist and title based on a fingerprint can also be seen as a static classification task (e.g., Wang 2003).

As the main aim and novelty of this thesis is to explore efficient large-scale, online feature extraction and propose ground-breaking standard feature sets, only the two most popular modelling methods Support-Vector Machines for the static case, and (Long Short-Term Memory) Recurrent Neural Networks for the dynamic case, are discussed in this section.

2.5.1 *Static Modelling with Support Vector Machines*

In static modelling a segment of audio is represented by a single, fixed dimensional vector of parameters (supra-segmental features, see Sect. 2.4) and modelled with a static classifier—usually a distance based, i.e., nearest neighbour or polynomial classifier, or a statistical classifier, such as a Bayes classifier (Kroschel et al. 2011). Due to the great success of SVMs in speech analysis for emotion recognition and Computational Paralinguistics (Schuller and Batliner 2013), this distance based polynomial classifier is the favoured classifier for static classification in this thesis.

SVMs—originally introduced by Cortes and Vapnik (1995)—are probably the most frequently used classifier for paralinguistic speech and music analysis tasks (Schuller 2013; Schuller and Batliner 2013) at present. This fact can be attributed to a handful of convenient properties—mainly their ability to handle high dimensional feature spaces (e.g., the baseline acoustic feature sets created for this thesis—cf. Chap. 3), noisy and sparse features (e.g., features which are almost always zero, such as frequently encountered in vector space modelling of linguistic features, cf. Schuller and Batliner 2013), and the robustness of their training algorithms to over-fitting. These properties have been discussed and pointed out by Joachims (1998) in the context of classification of texts with linguistic features, and they have subsequently been exploited for acoustic classification, too. Another advantage is that SVMs can be easily extended to continuous class labels (regression tasks) by Support Vector Regression (SVR) as introduced by Cortes and Vapnik (1995). In order to avoid repetition of material, which has been already described many times elsewhere, only a very coarse summary of the concept of linear SVM (as used in this thesis) is given at this point. For more details the reader is referred to the excellent summary of SVMs and SVR by Schuller and Batliner (2013) and the original paper by Cortes and Vapnik (1995).

The core idea of SVM is built around the concept of binary linear classifiers and is optimised towards providing the best possible separation between classes in the given feature space—which is the core difference to other classifiers with linear decision boundaries, such as the nearest neighbour classifiers. When training a SVM a hyperplane which separates the two classes in the feature space is constructed. In order to improve generalisation and limit over-fitting, a margin between the two classes which should be free of feature vectors is enforced and maximised during the construction of the hyperplane (Cortes and Vapnik 1995). Thereby a trade-off between margin maximisation and data points which then fall into the margin region must be found.

This optimisation criterion leads to a description of the hyperplane based on so called 'support vectors' which lie in between the centres of gravity of the classes and define the decision boundary (the hyperplane including a margin between the classes). The concept of the hyperplane and maximum margin is illustrated in Fig. 2.9.

The support vectors are chosen by solving a quadratic optimisation problem, for which efficient algorithms are available (cf. Cortes and Vapnik 1995; Platt 1998). As a result, the classification is based on a small subset of the training set of data points, effectively reducing the risk of over-fitting and improving generalisation performance. In order to solve non-linear decision tasks, i.e., tasks where the two classes cannot be separated by a linear decision border (hyperplane) in the feature space, the 'kernel trick' (Schölkopf and Smola 2002) is applied to map the non-linear problem into a higher dimensional space where it can be solved linearly in order to retain the low complexity of the (linear) support vector principle. For the

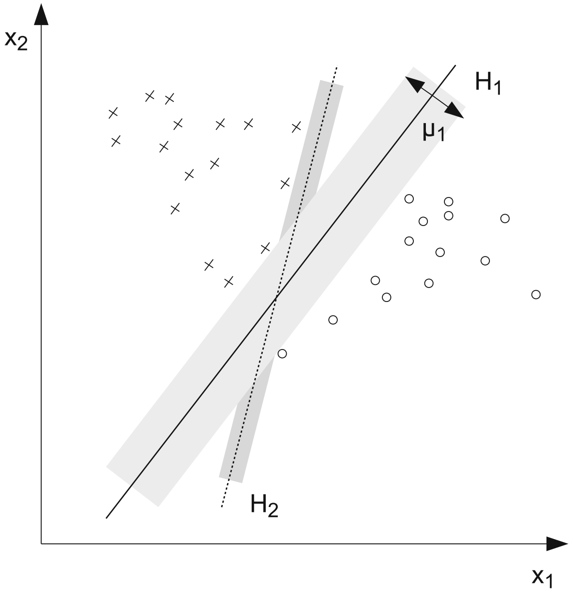

Fig. 2.9 Example of an (optimal) hyper plane H_1 (*solid line*) with maximum margin (μ_1) and a sub-optimal hype plane with smaller margin H_2 in a 2d feature space. The symbols "x" and "o" represent instances (data points) of the two classes, respectively (Schuller and Batliner 2013)

mathematics behind the training algorithms of support vector machines and support vector regression the summary by Schuller and Batliner (2013) can be consulted.

The SVMs as described so far are capable of discriminating between two classes only. For multi-class problems extensions must be built on top of the binary SVM. This could be, e.g., by building SVMs for each pair of classes, summing up the 'votes' for each class during recognition and then choosing the class with the most votes as winner, or by forming a binary decision tree (cf. Schuller and Batliner 2013) with each binary decision being performed by a SVM classification.

For linear kernel SVM, the hyperplane can be expressed in a compact representation by its normal vector. This makes linear kernel SVM highly suitable for real-time, on-line recognition tasks, as the decision function for an unknown instance can be computed as the scalar product of the normal vector and the feature vector of the unknown instance.

2.5.2 Dynamic Modelling

Dynamic modelling involves modelling of signal dynamics and context. In contrast to static modelling, where a single vector is mapped to a class or regression label, in dynamic modelling a sequence of feature vectors is mapped to a label or a sequence of labels. The modelling framework must be capable of modelling not only the associations of feature vectors to labels or label probabilities but also the dependencies between the labels in relation to the inputs (feature vectors) and the labels.

The most well known dynamic modelling technique in the field of ASR is HMMs (Rabiner and Juang 1986; Rabiner 1989). There, a statistical framework models the transitions between frames, while a probabilistic classifier models the data for each feature vector. For the latter, most commonly Gaussian Mixture Models (GMMs) are used. However, any classifier which is capable of returning an observation probability for a given feature vector could be used. Common examples are Neural Networks (NNs) (Stadermann and Rigoll 2006) and SVMs (Stadermann and Rigoll 2004). As HMMs are not a core topic of this thesis, the author refers to Rabiner (1989), Young et al. (2006), and Schuller and Batliner (2013) for more details on the concepts of HMMs for speech and music analysis.

An alternative dynamic modelling approach is based on neural networks. A standard Feed-Forward Neural Network (FFNN) can be extended to have access to data from previous timesteps. This extension is called Recurrent Neural Network (RNN) and will be described in the following section.

2.5.2.1 Recurrent Neural Networks

A standard FFNN is given by one or more hidden layers of sigmoid units. Each sigmoid unit (also referred to as *neuron*) consists of a weighted summation of inputs followed by a nonlinearity (Fig. 2.10, left). The output y of a sigmoid unit is described by the following equation:

$$y = g\left(b + \underline{\boldsymbol{w}} \cdot \underline{\boldsymbol{x}}\right) = g\left(b + \sum_{i=0}^{N-1} w_i x_i\right), \tag{2.327}$$

where N is the dimensionality of the input vector $\underline{\boldsymbol{x}}$ and the weight vector $\underline{\boldsymbol{w}}$ (both dimensions must match), and w_i and x_i are the ith elements of these vectors.

The function $g(.)$ can be any non-linear differentiable function in theory. Practically, however, the *sigmoid* function is used most often (Schuller 2013):

$$g_{sigmoid}(x) = \frac{1}{1 + e^{-\alpha x}}, \tag{2.328}$$

in its special case where $\alpha = 1$, which is known as the *logistic* function (Verhulst 1945). Alternatively, the hyperbolic tangent function tanh (Rade et al. 2000):

$$g_{tanh}(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}, \tag{2.329}$$

is used.

The purpose of the non-linearity is to **(a)** limit, i.e., compress, the output values and ensure stable outputs for a wide range of inputs, and—most important—**(b)** to enable the network to approximate any arbitrary non-linear function. For more details on the fundamentals of FFNNs, such as the training algorithms, the reader is referred to Bishop (1995).

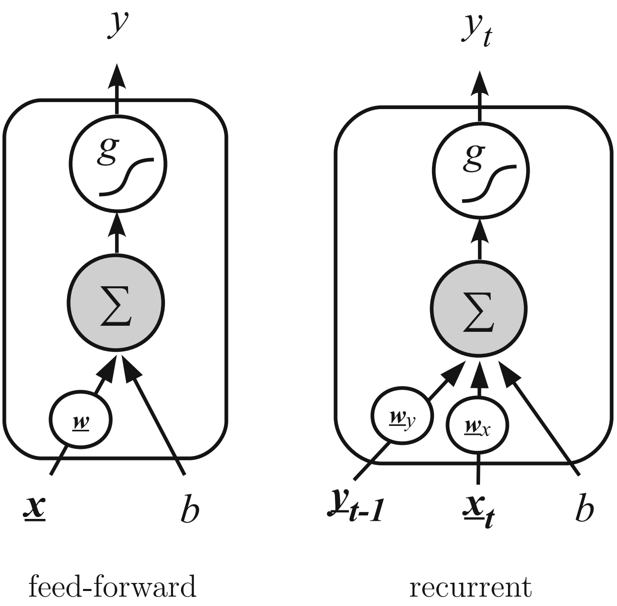

Fig. 2.10 Sigmoid neuron in a feedforward neural network (*left*) and sigmoid neuron in a recurrent neural network (*right*). Input vector $\underline{x}$, previous output (from all cells in the current layer) in vector $\underline{y}_{t-1}$, output activation y (scalar), constant bias b, and non-linearity $g(.)$

FFNNs with the standard sigmoid units have no memory, i.e., they have no knowledge of other inputs than those of the current timestep. A logical extension is to make the network recurrent, i.e., add a feedback from the output to the input with a delay of one timestep. Such networks are known as RNNs. In a RNN typically the outputs of all neurons of a recurrent hidden layer are connected back to the inputs of each of the neurons in that layer through a recurrent connection which has a time delay of one timestep. A single recurrent sigmoid unit is shown in Fig. 2.10 (right).

2.5.2.2 Long Short-Term Memory Recurrent Neural Networks

RNNs as introduced in the previous section, however, suffer from the Vanishing Gradient Problem (Hochreiter et al. 2001). This means that the activations and the backpropagated error on the recurrent connections (with weights ranging from $0 \ldots 1$) decay exponentially. This severely limits the amount of temporal context which is accessible to the networks effectively to approximately 10 frames. To overcome this problem, LSTM-RNNs have been proposed originally by Hochreiter and Schmidhuber (1997) and extended to the version used in this thesis by Graves and Schmidhuber (2005). The main difference between the original version and the version used here (and by Graves and Schmidhuber 2005) is the use of *peep-hole* connections (and according weights) from the internal cell memory state c_t to the input, output, and

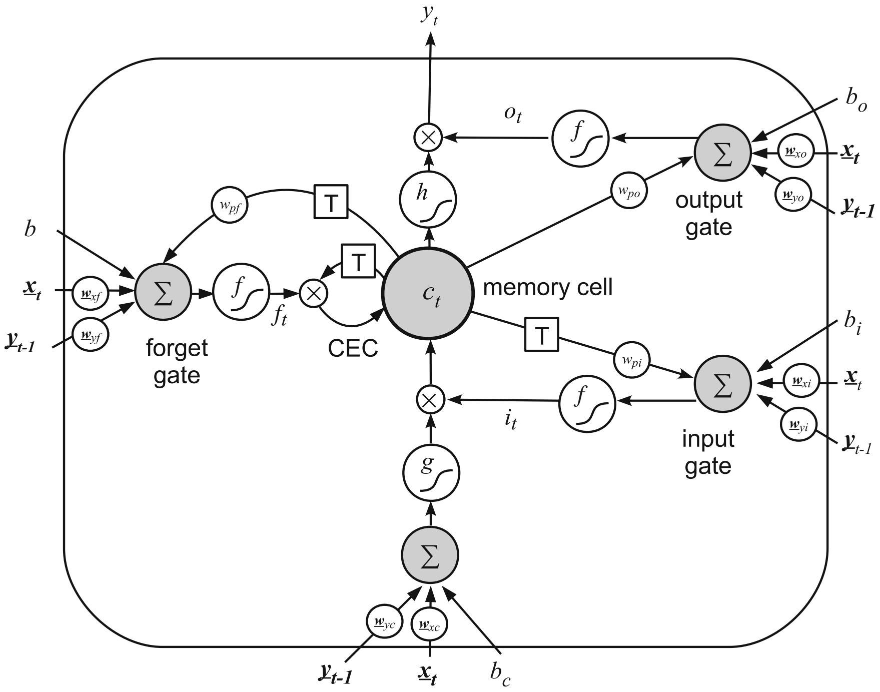
y_t
o_t
b_o
$\underline{\boldsymbol{w}}_{xo}$
$\underline{\boldsymbol{x}}_t$
$\underline{\boldsymbol{w}}_{yo}$
$\underline{\boldsymbol{y}}_{t-1}$
output gate
w_{po}
f
Σ
h
T
w_{pf}
b
$\underline{\boldsymbol{x}}_t$
$\underline{\boldsymbol{w}}_{xf}$
$\underline{\boldsymbol{y}}_{t-1}$
$\underline{\boldsymbol{w}}_{yf}$
Σ
f
f_t
T
forget gate
CEC
c_t
memory cell
T
w_{pi}
b_i
$\underline{\boldsymbol{w}}_{xi}$
$\underline{\boldsymbol{x}}_t$
$\underline{\boldsymbol{w}}_{yi}$
$\underline{\boldsymbol{y}}_{t-1}$
Σ
f
i_t
input gate
g
Σ
$\underline{\boldsymbol{w}}_{yc}$
$\underline{\boldsymbol{w}}_{xc}$
$\underline{\boldsymbol{y}}_{t-1}$
$\underline{\boldsymbol{x}}_t$
b_c

Fig. 2.11 Long Short-Term Memory block, with one LSTM cell and the input (i), output (o) and forget (f) gates. The block state is shown at timestep t. Input data vector ($\underline{\boldsymbol{x}}$), connection weights w_{ab} (multiplicative), bias values b, block output y. Vectors are indicated by *underlined bold face font*, all other variables are scalars. The vector containing all cell outputs y_t of the current hidden layer at timestep t is denoted as $\underline{\boldsymbol{y}}_t$. T denotes a time delay unit of one timestep/frame. X in a *circle* denotes a multiplicative unit. Σ denotes a summation unit. $f()$, $g()$, and $h()$ are non-linear activation functions (squashing functions)

forget gate summation units (cf. $w_{p..}$ in Fig. 2.11). Compared to a conventional RNN, the sigmoid summation units in the hidden layers are replaced by so-called Long Short-Term Memory (LSTM) blocks. LSTM blocks can theoretically store information in the cell variable c_t for an infinite amount of time due to the Constant Error Carousel (CEC) in which the previous cell state c_{t-1} is connected to the current state via a recurrent connection with a constant weight of 1 (excluding the multiplicative influence of the forget gate)—see Fig. 2.11. In this way, the network can dynamically exploit long-range temporal context present in the input data. In practice this ability has successfully been demonstrated on many speech and music analysis tasks, e.g., Wöllmer et al. (2008, 2010, 2013), Eyben et al. (2009b, 2010b), Böck and Schedl (2012), and Weninger et al. (2014).

Each LSTM block consists of a memory cell and three multiplicative gates: the input gate, the output gate, and the forget gate, as shown in Fig. 2.11. These gates control the access to the block's internal memory cell c_t. According to Fig. 2.11, the input, output and forget gate activation values i_t, o_t, and f_t are computed, respectively, as:

$$i_t = f(\underline{\boldsymbol{w}}_{xi}\underline{\boldsymbol{x}}_t + \underline{\boldsymbol{w}}_{yi}\underline{\boldsymbol{y}}_{t-1} + w_{pi}c_{t-1} + b_i), \tag{2.330}$$

$$o_t = f(\underline{\boldsymbol{w}}_{xo}\underline{\boldsymbol{x}}_t + \underline{\boldsymbol{w}}_{yo}\underline{\boldsymbol{y}}_{t-1} + w_{po}c_t + b_o), \tag{2.331}$$

$$f_t = f(\underline{\boldsymbol{w}}_{xf}\underline{\boldsymbol{x}}_t + \underline{\boldsymbol{w}}_{yf}\underline{\boldsymbol{y}}_{t-1} + w_{pf}c_{t-1} + b_f), \tag{2.332}$$

where $\underline{\boldsymbol{w}}_x$ and $\underline{\boldsymbol{w}}_y$ are weight vectors (row vectors) matching the dimensionality of $\underline{\boldsymbol{x}}$ or $\underline{\boldsymbol{y}}$, respectively. $\underline{\boldsymbol{x}}_t$ is the input vector at timestep t, $\underline{\boldsymbol{y}}_{t-1}$ is the vector of hidden layer activations (outputs of all N cells in the hidden layer) at the previous timestep, and $b_{i,o,f}$ denotes the respective input, output, or forget gate bias value for the cell.

The forget gate controls the decay of the stored input c_t. If $f_t = 0$, the previous cell state c_{t-1} is fully erased. The input and output gates are responsible for dynamically weighting the cell input and output, respectively. The cell state c_t at timestep t is expressed as:

$$c_t = f_t c_{t-1} + i_t g(\underline{\boldsymbol{w}}_{xc}\underline{\boldsymbol{x}}_t + \underline{\boldsymbol{w}}_{yc}\underline{\boldsymbol{y}}_{t-1} + b_c), \tag{2.333}$$

and the cell output is given as:

$$\boldsymbol{y}_t = o_t h(c_t). \tag{2.334}$$

The activation functions f (for the gates), g (for the input), and h (for the output) are non-linear squashing functions like those in normal sigmoid neurons. Common choices are the *logistic* function or the tanh function (Sect. 2.5.2.1). Recently, also Rectified Linear Units (ReLUs) have been proposed for standard (deep) NNs and for LSTM (Krizhevsky et al. 2012; Pham et al. 2013). While they give performance gains for (deep) NNs (Krizhevsky et al. 2012; Dahl et al. 2013), this was not reported for LSTM (Pham et al. 2013). Thus, the standard configuration as used by Graves and Schmidhuber (2005), for example, and most other related work[88] which uses LSTM and/or Bidirectional Long Short-Term Memory (BLSTM), is applied for this thesis: tanh activation functions for the cell input and output (functions g and h) and *logistic* activation functions for the gates (functions f).

In a LSTM-RNN a LSTM hidden layer consists of N LSTM blocks (Fig. 2.11) which are fully connected to all inputs and recurrently to all outputs of this layer as shown in Fig. 2.12.

In addition to LSTM-RNN, Bidirectional Long Short-Term Memory Recurrent Neural Networks (BLSTM-RNNs) (Schuster and Paliwal 1997) are employed in this thesis. A bidirectional recurrent network can access context from both past and future inputs, which makes it very suitable for processing data with de-synchronised inputs and outputs, or where the outputs (targets) have been centred at the middle (time-wise) of an input event. The bidirectional context is made possible by processing the data in both directions with two separate hidden layers, one processing the data sequence forward, the other backward. The output activations from both hidden layers are then fed to the same output layer, where they are fused. The combination of the

[88] E.g., as is also implemented in the CURRENNT toolkit (http://sourceforge.net/projects/currennt) and the RNNLIB (http://sourceforge.net/projects/rnnl/).

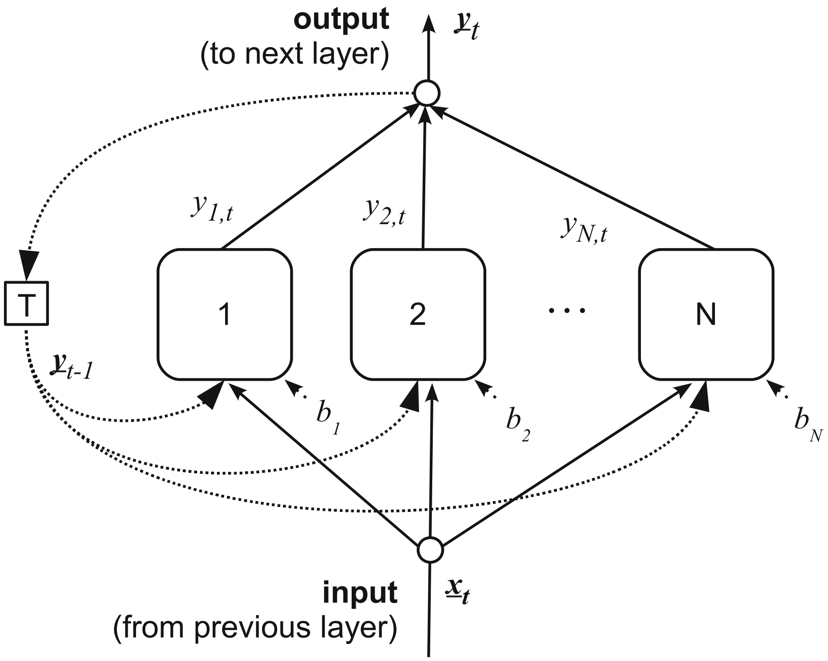

Fig. 2.12 Long Short-Term Memory hidden layer with *N* LSTM blocks, showing the fully connected recurrent connections of the outputs *y*

concept of bidirectional RNNs and LSTM blocks leads to BLSTM-RNNs (Graves and Schmidhuber 2005; Graves 2008).

LSTM-RNNs and BLSTM-RNNs as well as standard RNNs can be trained with a gradient descent method where the weights are iteratively updated, known from the backpropagation algorithm for FFNNs and extended to Backpropagation Through Time (BPTT) by Werbos (1990) for recurrent networks. A variation of the backpropagation algorithm is Resilient Propagation (rProp) introduced by Riedmiller and Braun (1993). There, only the sign of the error gradient is backpropagated and used for weight updates instead of the absolute value of the error weighted by the learn rate. Resilient propagation produces more stable convergence (e.g., Eyben et al. 2010c) and thus can outperform standard backpropagation, especially with respect to the number of training epochs required.

A more detailed summary of BLSTM-RNNs and the training algorithms is found in Schuller (2013) and an extensive discussion is found in the PhD thesis of Graves (2008).

References

R.G. Bachu, S. Kopparthi, B. Adapa, B.D. Barkana, Voiced/unvoiced decision for speech signals based on zero-crossing rate and energy, in *Advanced Techniques in Computing Sciences and Software Engineering*, ed. by K. Elleithy (Springer, Netherlands, 2010), pp. 279–282. doi:10.1007/978-90-481-3660-5_47. ISBN 978-90-481-3659-9

A. Batliner, S. Steidl, B. Schuller, D. Seppi, T. Vogt, L. Devillers, L. Vidrascu, N. Amir, L. Kessous, V. Aharonson, The impact of F0 extraction errors on the classification of prominence and emotion, in *Proceedings of 16-th ICPhS* (Saarbrücken, Germany, 2007), pp. 2201–2204

L.L. Beranek, *Acoustic Measurements* (Wiley, New York, 1949)

C.M. Bishop, *Neural Networks for Pattern Recognition* (Oxford University Press, New York, 1995)

R.B. Blackman, J. Tukey, Particular pairs of windows, *The Measurement of Power Spectra, from the Point of View of Communications Engineering* (Dover, New York, 1959)

S. Böck, M. Schedl, Polyphonic piano note transcription with recurrent neural networks, in *Proceedings of ICASSP 2012* (Kyoto, 2012), pp. 121–124

P. Boersma, Accurate short-term analysis of the fundamental frequency and the harmonics-to-noise ratio of a sampled sound. IFA Proc. **17**, 97–110 (1993)

P. Boersma, Praat, a system for doing phonetics by computer. Glot Int. **5**(9/10), 341–345 (2001)

B.P. Bogert, M.J.R. Healy, J.W. Tukey, The quefrency alanysis of time series for echoes: cepstrum, pseudo-autocovariance, cross-cepstrum, and saphe cracking, in *Proceedings of the Symposium on Time Series Analysis, chapter 15*, ed. by M. Rosenblatt (Wiley, New York, 1963), pp. 209–243

C.H. Chen, *Signal Processing Handbook*. Electrical Computer Engineering, vol. 51 (CRC Press, New York, 1988), 840 p. ISBN 978-0824779566

A. Cheveigne, H. Kawahara, YIN, a fundamental frequency estimator for speech and music. J. Acoust. Soc. Am. (JASA) **111**(4), 1917–1930 (2002)

T.-S. Chi, L.-Y. Yeh, C.-C. Hsu, Robust emotion recognition by spectro-temporal modulation statistic features. J. Ambient Intell. Humaniz. Comput. **3**, 47–60 (2012). doi:10.1007/s12652-011-0088-5

J. Cooley, P. Lewis, P. Welch, The finite fourier transform. IEEE Trans. Audio Electroacoust. **17**(2), 77–85 (1969)

C. Cortes, V. Vapnik, Support-vector networks. Mach. Learn. **20**(3), 273–297 (1995)

R. Cowie, E. Douglas-Cowie, S. Savvidou, E. McMahon, M. Sawey, M. Schröder, Feeltrace: an instrument for recording perceived emotion in real time, in *Proceedings of the ISCA Workshop on Speech and Emotion* (Newcastle, Northern Ireland, 2000), pp. 19–24

G. Dahl, T. Sainath, G. Hinton, Improving deep neural networks for LVCSR using rectified linear units and dropout, in *Proceedings of ICASSP 2013* (IEEE, Vancouver, 2013), pp. 8609–8613

G. Dalquist, A. Björk, N. Anderson, *Numerical Methods* (Prentice Hall, Englewood Cliffs, 1974)

S. Damelin, W. Miller, *The Mathematics of Signal Processing* (Cambridge University Press, Cambridge, 2011). ISBN 978-1107601048

G. de Krom, A cepstrum-based technique for determining a harmonics-to-noise ratio in speech signals. J. Speech Hear. Res. **36**, 254–266 (1993)

J.R. Deller, J.G. Proakis, J.H.L. Hansen, *Discrete-Time Processing of Speech Signals*, University of Michigan, Macmillan Publishing Company (1993)

P. Deuflhard, *Newton Methods For Nonlinear Problems: Affine Invariance and Adaptive Algorithms*. Springer Series in Computational Mathematics, vol. 35 (Springer, Berlin, 2011), 440 p

E. Douglas-Cowie, R. Cowie, I. Sneddon, C. Cox, O. Lowry, M. McRorie, J.C. Martin, L. Devillers, S. Abrilian, A. Batliner, N. Amir, K. Karpouzis, *The HUMAINE Database*. Lecture Notes in Computer Science, vol. 4738 (Springer, Berlin, 2007), pp. 488–500

J. Durbin, The fitting of time series models. Revue de l'Institut International de Statistique (Review of the International Statistical Institute) **28**(3), 233–243 (1960)

C. Duxbury, M. Sandler, M. Davies, A hybrid approach to musical note onset detection, in *Proceedings of the Digital Audio Effect Conference (DAFX'02)* (Hamburg, Germany, 2002), pp. 33–38

L.D. Enochson, R.K. Otnes, *Programming and Analysis for Digital Time Series Data*, 1st edn. U.S. Department of Defense, Shock and Vibration Information Center (1968)

F. Eyben, B. Schuller, Music classification with the Munich openSMILE toolkit, in *Proceedings of the Annual Meeting of the MIREX 2010 community as part of the 11th International Conference on Music Information Retrieval (ISMIR)* (ISMIR, Utrecht, 2010a). http://www.music-ir.org/mirex/abstracts/2010/FE1.pdf

F. Eyben, B. Schuller, Tempo estimation from tatum and meter vectors, in *Proceedings of the Annual Meeting of the MIREX 2010 community as part of the 11th International Conference on Music Information Retrieval (ISMIR)* (ISMIR, Utrecht, 2010b). www.music-ir.org/mirex/abstracts/2010/ES1.pdf

F. Eyben, M. Wöllmer, B. Schuller, openEAR—introducing the Munich open-source emotion and affect recognition toolkit, in *Proceedings of the 3rd International Conference on Affective Computing and Intelligent Interaction (ACII 2009)*, vol. I (IEEE, Amsterdam, 2009a), pp. 576–581

F. Eyben, M. Wöllmer, B. Schuller, A. Graves, From speech to letters—using a novel neural network architecture for grapheme based ASR, in *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU) 2009* (IEEE, Merano, 2009b), pp. 376–380

F. Eyben, M. Wöllmer, B. Schuller, openSMILE—The Munich versatile and fast open-source audio feature extractor, in *Proceedings of ACM Multimedia 2010* (ACM, Florence, 2010a), pp. 1459–1462

F. Eyben, S. Böck, B. Schuller, A. Graves, Universal onset detection with bidirectional long-short term memory neural networks, in *Proceedings of ISMIR 2010* (ISMIR, Utrecht, The Netherlands, 2010b), pp. 589–594

F. Eyben, M. Wöllmer, A. Graves, B. Schuller, E. Douglas-Cowie, R. Cowie, On-line emotion recognition in a 3-D activation-valence-time continuum using acoustic and linguistic cues. J. Multimodal User Interfaces (JMUI) **3**(1–2), 7–19 (2010c). doi:10.1007/s12193-009-0032-6

F. Eyben, M. Wöllmer, B. Schuller, A multi-task approach to continuous five-dimensional affect sensing in natural speech, ACM Trans. Interact. Intell. Syst. **2**(1), Article No. 6, 29 p. Special Issue on Affective Interaction in Natural Environments (2012)

G. Fant, *Speech Sounds and Features* (MIT press, Cambridge, 1973), p. 227

H.G. Feichtinger, T. Strohmer, *Gabor Analysis and Algorithms* (Birkhäuser, Boston, 1998). ISBN 0-8176-3959-4

J.-B.-J. Fourier, Théorie analytique de la chaleur, University of Lausanne, Switzerland (1822)

T. Fujishima, Realtime chord recognition of musical sound: a system using common lisp music, in *Proceedings of the International Computer Music Conference (ICMC) 1999* (Bejing, China, 1999), pp. 464–467

S. Furui, *Digital Speech Processing: Synthesis, and Recognition*. Signal Processing and Communications, 2nd edn. (Marcel Denker Inc., New York, 1996)

C. Glaser, M. Heckmann, F. Joublin, C. Goerick, Combining auditory preprocessing and bayesian estimation for robust formant tracking. IEEE Trans. Audio Speech Lang. Process. **18**(2), 224–236 (2010)

E. Gómez, Tonal description of polyphonic audio for music content processing. INFORMS J. Comput. **18**(3), 294–304 (2006). doi:10.1287/ijoc.1040.0126

F. Gouyon, F. Pachet, O. Delerue. Classifying percussive sounds: a matter of zero-crossing rate? in *Proceedings of the COST G-6 Conference on Digital Audio Effects (DAFX-00)* (Verona, Italy, 2000)

A. Graves, Supervised sequence labelling with recurrent neural networks. Doctoral thesis, Technische Universität München, Munich, Germany (2008)

A. Graves, J. Schmidhuber, Framewise phoneme classification with bidirectional LSTM and other neural network architectures. Neural Netw. **18**(5–6), 602–610 (2005)

W.D. Gregg, *Analog & Digital Communication* (Wiley, New York, 1977). ISBN 978-0-471-32661-8

M. Grimm, K. Kroschel, S. Narayanan, Support vector regression for automatic recognition of spontaneous emotions in speech, in *Proceedings of ICASSP 2007*, vol. 4 (IEEE, Honolulu, 2007), pp. 1085–1088

B. Hammarberg, B. Fritzell, J. Gauffin, J. Sundberg, L. Wedin, Perceptual and acoustic correlates of abnormal voice qualities. Acta Otolaryngol. **90**, 441–451 (1980)

H. Hanson, Glottal characteristics of female speakers: acoustic correlates. J. Acoust. Soc. Am. (JASA) **101**, 466–481 (1997)

H. Hanson, E.S. Chuang, Glottal characteristics of male speakers: acoustic correlates and comparison with female data. J. Acoust. Soc. Am. (JASA) **106**, 1064–1077 (1999)

F.J. Harris, On the use of windows for harmonic analysis with the discrete fourier transform. Proc. IEEE **66**, 51–83 (1978)

H. Hermansky, Perceptual linear predictive (PLP) analysis for speech. J. Acoust. Soc. Am. (JASA) **87**, 1738–1752 (1990)

H. Hermansky, N. Morgan, A. Bayya, P. Kohn, RASTA-PLP speech analysis technique, in *Proceedings of ICASSP 1992*, vol. 1 (IEEE, San Francisco, 1992), pp. 121–124

D.J. Hermes, Measurement of pitch by subharmonic summation. J. Acoust. Soc. Am. (JASA) **83**(1), 257–264 (1988)

W. Hess, *Pitch Determination of Speech Signals: Algorithms and Devices* (Springer, Berlin, 1983)

S. Hochreiter, J. Schmidhuber, Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)

S. Hochreiter, Y. Bengio, P. Frasconi, J. Schmidhuber, Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, in *A Field Guide to Dynamical Recurrent Neural Networks*, ed. by S.C. Kremer, J.F. Kolen (IEEE Press, New York, 2001)

ISO16:1975. ISO Standard 16:1975 Acoustics: Standard tuning frequency (Standard musical pitch). International Organization for Standardization (ISO) (1975)

T. Joachims, Text categorization with support vector machines: learning with many relevant features, in *Proceedings of the 10th European Conference on Machine Learning (ECML-98)*, ed. by C. Nédellec, C. Rouveirol (Springer, Chemnitz, 1998), pp. 137–142

J.D. Johnston, Transform coding of audio signals using perceptual noise criteria. IEEE J. Sel. Areas Commun. **6**(2), 314–332 (1988)

P. Kabal, R.P. Ramachandran, The computation of line spectral frequencies using Chebyshev polynomials. IEEE Trans. Acoust. Speech Signal Process. **34**(6), 1419–1426 (1986)

J.F. Kaiser, Some useful properties of teager's energy operators, in *Proceedings of ICASSP 1993*, vol. 3, pp. 149–152, (IEEE, Minneapolis, 1993). doi:10.1109/ICASSP.1993.319457

G.S. Kang, L.J. Fransen, Application of line spectrum pairs to low bit rate speech encoders, in *Proceedings of ICASSP 1985*, vol.10 (IEEE, Tampa, 1985), pp. 244–247. doi:10.1109/ICASSP.1985.1168526

R. Kendall, E. Carterette, Difference thresholds for timbre related to spectral centroid, in *Proceedings of the 4-th International Conference on Music Perception and Cognition (ICMPC)* (Montreal, Canada, 1996), pp. 91–95

J.F. Kenney, E.S. Keeping, Root mean square, *Mathematics of Statistics*, vol. 1, 3rd edn. (Van Nostrand, Princeton, 1962), pp. 59–60

A. Khintchine, Korrelationstheorie der stationären stochastischen prozesse. Math. Ann. **109**, 604–615 (1934)

A. Kießling, *Extraktion und Klassifikation prosodischer Merkmale in der automatischen Sprachverarbeitung* (Shaker, Aachen, 1997). ISBN 978-3-8265-2245-1

A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in *Advances in Neural Information Processing Systems*, vol. 25, ed. by F. Pereira, C.J.C. Burges, L. Bottou, K.Q. Weinberger (Curran Associates, Inc., 2012), pp. 1097–1105

K. Kroschel, G. Rigoll, B. Schuller, *Statistische Informationstechnik*, 5th edn. (Springer, Berlin, 2011)

K. Lee, M. Slaney, Acoustic chord transcription and key extraction from audio using key-dependent HMMs trained on synthesized audio. IEEE Trans. Audio Speech Lang. Process. **16**(2), 291–301 (2008). doi:10.1109/TASL.2007.914399. ISSN 1558-7916

P. Lejeune-Dirichlet, Sur la convergence des séries trigonométriques qui servent à représenter une fonction arbitraire entre des limites données. Journal für die reine und angewandte Mathematik **4**, 157–169 (1829)

N. Levinson, A heuristic exposition of wiener's mathematical theory of prediction and filtering. J. Math. Phys. **25**, 110–119 (1947a)

N. Levinson, The Wiener RMS error criterion in filter design and prediction. J. Math. Phys. **25**(4), 261–278 (1947b)

P.I. Lizorkin, Fourier transform, in *Encyclopaedia of Mathematics*, ed. by M. Hazewinkel (Springer, Berlin, 2002). ISBN 1-4020-0609-8

I. Luengo, Evaluation of pitch detection algorithms under real conditions, in *Proceedings of ICASSP 2007*, vol. 4 (IEEE, Honolulu, 2007), pp. 1057–1060

J. Makhoul, Linear prediction: a tutorial review. Proc. IEEE **63**(5), 561–580 (1975)

J. Makhoul, L. Cosell, LPCW: an LPC vocoder with linear predictive spectral warping, in *Proceedings of ICASSP 1976* (IEEE, Philadelphia, 1976), pp. 466–469

B.S. Manjunath, P. Salembier, T. Sikoraa (eds.), *Introduction to MPEG-7: Multimedia Content Description Interface* (Wiley, Berlin, 2002), 396 p. ISBN 978-0-471-48678-7

P. Martin, Détection de f_0 par intercorrelation avec une fonction peigne. J. Etude Parole **12**, 221–232 (1981)

P. Martin, Comparison of pitch detection by cepstrum and spectral comb analysis, in *Proceedings of ICASSP 1982* (IEEE, Paris, 1982), pp. 180–183

J. Martinez, H. Perez, E. Escamilla, M.M. Suzuki, Speaker recognition using mel frequency cepstral coefficients (MFCC) and vector quantization (VQ) techniques, in *Proceedings of the 22nd International Conference on Electrical Communications and Computers (CONIELECOMP)* (Cholula, Puebla, 2012), pp. 248–251. doi:10.1109/CONIELECOMP.2012.6189918

P. Masri, Computer modelling of sound for transformation and synthesis of musical signal. Doctoral thesis, University of Bristol, Bristol (1996)

S. McCandless, An algorithm for automatic formant extraction using linear prediction spectra. IEEE Trans. Acoust. Speech Signal Process. **22**, 134–141 (1974)

D.D. Mehta, D. Rudoy, P.K. Wolfe, Kalman-based autoregressive moving average modeling and inference for formant and antiformant tracking. J. Acoust. Soc. Am. (JASA) **132**(3), 1732–1746 (2012)

H. Misra, S. Ikbal, H. Bourlard, H. Hermansky, Spectral entropy based feature for robust ASR, in *Proceedings of ICASSP 2004*, vol. 1 (IEEE, Montreal, Canada, 2004), pp. I–193–6. doi:10.1109/ICASSP.2004.1325955

O. Mubarak, E. Ambikairajah, J. Epps, T. Gunawan, Modulation features for speech and music classification, in *Proceedings of the 10th IEEE Singapore International Conference on Communication systems (ICCS) 2006* (IEEE, 2006), pp. 1–5. doi:10.1109/ICCS.2006.301515

M. Müller, *Information Retrieval for Music and Motion* (Springer, Berlin, 2007)

M. Müller, F. Kurth, M. Clausen, Audio matching via chroma-based statistical features, in *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR)* (London, 2005a), pp. 288–295

M. Müller, F. Kurth, M. Clausen, Chroma-based statistical audio features for audio matching, in *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics* (IEEE, 2005b), pp. 275–278

N.J. Nalini, S. Palanivel, Emotion recognition in music signal using AANN and SVM. Int. J. Comput. Appl. **77**(2), 7–14 (2013)

A.M. Noll, Cepstrum pitch determination. J. Acoust. Soc. Am. (JASA) **41**(2), 293–309 (1967)

A.M. Noll, Pitch determination of human speech by the harmonic product spectrum, the harmonic sum spectrum, and a maximum likelihood estimate, in *Symposium on Computer Processing in Communication*, vol. 19 (University of Brooklyn, New York, 1970), pp. 779–797, edited by the Microwave Institute

A.H. Nuttal, Some windows with very good sidelobe behavior. IEEE Trans. Acoust. Speech Signal Process. ASSP **29**, 84–91 (1981)

A.V. Oppenheim, R.W. Schafer, *Digital Signal Processing* (Prentice-Hall, Englewood Cliffs, 1975)

A.V. Oppenheim, A.S. Willsky, S. Hamid, *Signals and Systems*, 2nd edn. (Prentice Hall, Upper Saddle River, 1996)

A.V. Oppenheim, R.W. Schafer, J.R. Buck, *Discrete-Time Signal Processing* (Prentice Hall, Upper Saddle River, 1999)

T.W. Parsons, *Voice and Speech Processing*. Electrical and Computer Engineering (University of Michigan, McGraw-Hill, 1987)

S. Patel, K.R. Scherer, J. Sundberg, E. Björkner, Acoustic markers of emotions based on voice physiology, in *Proceedings of Speech Prosody 2010* (ISCA, Chicago, 2010), pp. 100865:1–4

G. Peeters, A large set of audio features for sound description. Technical report, IRCAM, Switzerland (2004). http://recherche.ircam.fr/equipes/analyse-synthese/peeters/ARTICLES/Peeters_2003_cuidadoaudiofeatures.pdf. Accessed 3 Sept. 2013

V. Pham, C. Kermorvant, J. Louradour, Dropout improves recurrent neural networks for handwriting recognition, in *CoRR* (2013) (online), arXiv:1312.4569

J. Platt, Sequential minimal optimization: a fast algorithm for training support vector machines, Technical report MSR-98-14, Microsoft Research (1998)

L.R. Rabiner, On the use of autocorrelation analysis for pitch detection. IEEE Trans. Acoust. Speech Signal Process. **25**(1), 24–33 (1977). doi:10.1109/TASSP.1977.1162905

L.R. Rabiner, A tutorial on hidden markov models and selected applications in speech recognition. Proc. IEEE **77**(2), 257–286 (1989)

L.R. Rabiner, B.H. Juang, An introduction to hidden markov models. IEEE ASSP Mag. **3**(1), 4–16 (1986)

L. Rabiner, B.-H. Juang, *Fundamentals of Speech Recognition*, 1st edn. (Prentice Hall, Englewood Cliffs, 1993)

L. Rade, B. Westergren, *Springers Mathematische Formeln (German translation by P. Vachenauer)*, 3rd edn. (Springer, Berlin, 2000). ISBN 3-540-67505-1

J.F. Reed, F. Lynn, B.D. Meade, Use of coefficient of variation in assessing variability of quantitative assays. Clin. Diagn. Lab. Immunol. **9**(6), 1235–1239 (2002)

M. Riedmiller, H. Braun, A direct adaptive method for faster backpropagation learning: the RPROP algorithm, in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 1 (IEEE, San Francisco, 1993), pp. 586–591. doi:10.1109/icnn.1993.298623

F. Ringeval, A. Sonderegger, J. Sauer, D. Lalanne, Introducing the RECOLA multimodal corpus of remote collaborative and affective interactions, in *Proceedings of the 2nd International Workshop on Emotion Representation, Analysis and Synthesis in Continuous Time and Space (EmoSPACE), held in conjunction with FG 2013* (IEEE, Shanghai, 2013), pp. 1–8

S. Rosen, P. Howell, The vocal tract as a linear system, *Signals and Systems for Speech and Hearing*, 1st edn. (Emerald Group, 1991), pp. 92–99. ISBN 978-0125972314

G. Ruske, *Automatische Spracherkennung*. Methoden der Klassifikation und Merkmalsextraktion, 2nd edn. (Oldenbourg, Munich, 1993)

K.R. Scherer, J. Sundberg, L. Tamarit, G.L. Salomão, Comparing the acoustic expression of emotion in the speaking and the singing voice. Comput. Speech Lang. **29**(1), 218–235 (2015). doi:10.1016/j.csl.2013.10.002

B. Schölkopf, A. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond (Adaptive Computation and Machine Learning)* (MIT Press, Cambridge, 2002)

M. Schröder, E. Bevacqua, R. Cowie, F. Eyben, H. Gunes, D. Heylen, M. ter Maat, G. McKeown, S. Pammi, M. Pantic, C. Pelachaud, B. Schuller, E. de Sevin, M. Valstar, M. Wöllmer, Building autonomous sensitive artificial listeners. IEEE Trans. Affect. Comput. **3**(2), 165–183 (2012)

M.R. Schroeder, Period histogram and product spectrum: new methods for fundamental-frequency measurement. J. Acoust. Soc. Am. (JASA) **43**, 829–834 (1968)

M.R. Schroeder, Recognition of complex acoustic signals, in *Life Sciences Research Reports*, vol. 5, ed. by T.H. Bullock (Abakon Verlag, Berlin, 1977), 324 p

B. Schuller, Automatische Emotionserkennung aus sprachlicher und manueller Interaktion. Doctoral thesis, Technische Universität München, Munich, Germany (2006)

B. Schuller, *Intelligent Audio Analysis*. Signals and Communication Technology (Springer, Berlin, 2013)

B. Schuller, A. Batliner, *Computational Paralinguistics: Emotion, Affect and Personality in Speech and Language Processing* (Wiley, Hoboken, 2013), 344 p. ISBN 978-1119971368

B. Schuller, G. Rigoll, M. Lang, Hidden Markov model-based speech emotion recognition, in *Proceedings of ICASSP 2003*, vol. 2 (IEEE, Hong Kong, 2003), pp. II 1–4

B. Schuller, D. Arsić, F. Wallhoff, G. Rigoll, Emotion recognition in the noise applying large acoustic feature sets, in *Proceedings of the 3rd International Conference on Speech Prosody (SP) 2006* (ISCA, Dresden, 2006), pp. 276–289

B. Schuller, F. Eyben, G. Rigoll, Fast and robust meter and tempo recognition for the automatic discrimination of ballroom dance styles, in *Proceedings of ICASSP 2007*, vol. I (IEEE, Honolulu, 2007), pp. 217–220

B. Schuller, F. Eyben, G. Rigoll, Beat-synchronous data-driven automatic chord labeling, in *Proceedings 34. Jahrestagung für Akustik (DAGA) 2008* (DEGA, Dresden, 2008), pp. 555–556

B. Schuller, S. Steidl, A. Batliner, F. Jurcicek, The INTERSPEECH 2009 emotion challenge, in *Proceedings of INTERSPEECH 2009* (Brighton, 2009a), pp. 312–315

B. Schuller, B. Vlasenko, F. Eyben, G. Rigoll, A. Wendemuth, Acoustic emotion recognition: A benchmark comparison of performances, in *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU) 2009* (IEEE, Merano, 2009b), pp. 552–557

B. Schuller, S. Steidl, A. Batliner, F. Burkhardt, L. Devillers, C. Müller, S. Narayanan, The INTERSPEECH 2010 paralinguistic challenge, in *Proceedings of INTERSPEECH 2010* (ISCA, Makuhari, 2010), pp. 2794–2797

B. Schuller, A. Batliner, S. Steidl, F. Schiel, J. Krajewski, The INTERSPEECH 2011 speaker state challenge, in *Proceedings of INTERSPEECH 2011* (ISCA, Florence, 2011), pp. 3201–3204

B. Schuller, S. Steidl, A. Batliner, E. Nöth, A. Vinciarelli, F. Burkhardt, R. van Son, F. Weninger, F. Eyben, T. Bocklet, G. Mohammadi, B. Weiss, The INTERSPEECH 2012 speaker trait challenge, in *Proceedings of INTERSPEECH 2012* (ISCA, Portland, 2012a)

B. Schuller, M. Valstar, R. Cowie, M. Pantic, AVEC 2012: the continuous audio/visual emotion challenge—an introduction, in *Proceedings of the 14th ACM International Conference on Multimodal Interaction (ICMI) 2012*, ed. by L.-P. Morency, D. Bohus, H.K. Aghajan, J. Cassell, A. Nijholt, J. Epps (ACM, Santa Monica, 2012b), pp. 361–362. October

B. Schuller, F. Pokorny, S. Ladstätter, M. Fellner, F. Graf, L. Paletta. Acoustic geo-sensing: recognising cyclists' route, route direction, and route progress from cell-phone audio, in *Proceedings of ICASSP 2013* (IEEE, Vancouver, 2013a), pp. 453–457

B. Schuller, S. Steidl, A. Batliner, A. Vinciarelli, K. Scherer, F. Ringeval, M. Chetouani, et al., The INTERSPEECH 2013 computational paralinguistics challenge: social signals, conflict, emotion, autism, in *Proceedings of INTERSPEECH 2013* (ISCA, Lyon, 2013b), pp. 148–152

M. Schuster, K.K. Paliwal, Bidirectional recurrent neural networks. IEEE Trans. Signal Process. **45**(11), 2673–2681 (1997)

C.E. Shannon, A mathematical theory of communication. Bell Syst. Tech. J. **27**, 379–423, 623–656 (1948). (Reprint with corrections in: ACM SIGMOBILE Mobile Computing and Communications Review **5**(1), 3–55 (2001))

M. Slaney, An efficient implementation of the patterson-holdsworth auditory filter bank. Technical Report 35, Apple Computer Inc. (1993)

M. Soleymani, M.N. Caro, E.M. Schmidt, Y.-H. Yang, The MediaEval 2013 brave new task: emotion in music, in *Proceedings of the MediaEval 2013 Workshop* (CEUR-WS.org, Barcelona, 2013)

F.K. Soong, B.-W. Juang, Line spectrum pair (LSP) and speech data compression, in *Proceedings of ICASSP 1984* (IEEE, San Diego, 1984), pp. 1.10.1–1.10.4

A. Spanias, T. Painter, V. Atti, *Audio Signal Processing and Coding* (Wiley, Hoboken, 2007), 464 p. ISBN 978-0-471-79147-8

J. Stadermann, G. Rigoll, A hybrid SVM/HMM acoustic modeling approach to automatic speech recognition, in *Proceedings of INTERSPEECH 2004* (ISCA, Jeju, 2004), pp. 661–664

J. Stadermann, G. Rigoll, Hybrid NN/HMM acoustic modeling techniques for distributed speech recognition. Speech Commun. **48**(8), 1037–1046 (2006)

J.F. Steffensen, *Interpolation*, 2nd edn. (Dover Publications, New York, 2012), 256 p. ISBN 978-0486154831

P. Suman, S. Karan, V. Singh, R. Maringanti, Algorithm for gunshot detection using mel-frequency cepstrum coefficients (MFCC), in *Proceedings of the Ninth International Conference on Wireless Communication and Sensor Networks*, ed. by R. Maringanti, M. Tiwari, A. Arora. Lecture Notes in Electrical Engineering, vol. 299 (Springer, India, 2014), pp. 155–166. doi:10.1007/978-81-322-1823-4_15. ISBN 978-81-322-1822-7

J. Sundberg, *The Science of the Singing Voice* (Northern Illinois University Press, Dekalb, 1987), p. 226. ISBN 978-0-87580-542-9

D. Talkin, A robust algorithm for pitch tracking (RAPT), in *Speech Coding and Synthesis*, ed. by W.B. Kleijn, K.K. Paliwal (Elsevier, New York, 1995), pp. 495–518. ISBN 0444821694

L. Tamarit, M. Goudbeek, K.R. Scherer, Spectral slope measurements in emotionally expressive speech, in *Proceedings of SPKD-2008* (ISCA, 2008), paper 007

H.M. Teager, S.M. Teager, Evidence for nonlinear sound production mechanisms in the vocal tract, in *Proceedings of Speech Production and Speech Modelling, Bonas, France*, ed. by W.J. Hardcastle, A. Marchal. NATO Advanced Study Institute Series D, vol. 55 (Kluwer Academic Publishers, Boston, 1990), pp. 241–261

E. Terhardt, Pitch, consonance, and harmony. J. Acoust. Soc. Am. (JASA) **55**, 1061–1069 (1974)

E. Terhardt, Calculating virtual pitch. Hear. Res. **1**, 155–182 (1979)

H. Traunmueller, Analytical expressions for the tonotoc sensory scale. J. Acoust. Soc. Am. (JASA) **88**, 97–100 (1990)

K. Turkowski, S. Gabriel, Filters for common resampling tasks, in *Graphics Gems*, ed. by A.S. Glassner (Academic Press, New York, 1990), pp. 147–165. ISBN 978-0-12-286165-9

G. Tzanetakis, P. Cook, Musical genre classification of audio signals. IEEE Trans. Speech Audio Process. **10**(5), 293–302 (2002). doi:10.1109/TSA.2002.800560. ISSN 1063-6676

P.-F. Verhulst, Recherches mathématiques sur la loi d'accroissement de la population (mathematical researches into the law of population growth increase). Nouveaux Mémoires de l'Académie Royale des Sciences et Belles-Lettres de Bruxelles **18**, 1–42 (1945)

D. Ververidis, C. Kotropoulos, Emotional speech recognition: resources, features, and methods. Speech Commun. **48**(9), 1162–1181 (2006)

A.J. Viterbi, Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. IEEE Trans. Inf. Theory **13**(2), 260–269 (1967)

B. Vlasenko, B. Schuller, A. Wendemuth, G. Rigoll., Frame vs. turn-level: emotion recognition from speech considering static and dynamic processing, in *Proceedings of the 2nd International Conference on Affective Computing and Intelligent Interaction (ACII) 2007*, ed. by A. Paiva, R. Prada, R.W. Picard. Lecture Notes in Computer Science, Lisbon, Portugal, vol. 4738 (Springer, Berlin, 2007), pp. 139–147

A.L. Wang, An industrial-strength audio search algorithm, in *Proceedings of ISMIR* (Baltimore, 2003)

F. Weninger, F. Eyben, B. Schuller, The TUM approach to the mediaeval music emotion task using generic affective audio features, in *Proceedings of the MediaEval 2013 Workshop* (CEUR-WS.org, Barcelona, 2013)

F. Weninger, F. Eyben, B. Schuller, On-line continuous-time music mood regression with deep recurrent neural networks, in *Proceedings of ICASSP 2014* (IEEE, Florence, 2014), pp. 5449–5453

P. Werbos, Backpropagation through time: what it does and how to do it. Proc. IEEE **78**, 1550–1560 (1990)

N. Wiener, Generalized harmonic analysis. Acta Math. **55**(1), 117–258 (1930)

N. Wiener, *Extrapolation, Intrapolation and Smoothing of Stationary Time Series*, M.I.T. Press Paperback Series (Book 9) (MIT Press, Cambridge, 1964), 163 p

M. Wöllmer, F. Eyben, S. Reiter, B. Schuller, C. Cox, E. Douglas-Cowie, R. Cowie, Abandoning emotion classes—towards continuous emotion recognition with modelling of long-range dependencies, in *Proceedings of INTERSPEECH 2008* (ISCA, Brisbane, 2008), pp. 597–600

M. Wöllmer, F. Eyben, A. Graves, B. Schuller, G. Rigoll, Improving keyword spotting with a tandem BLSTM-DBN architecture, in *Advances in Non-linear Speech Processing: Revised selected papers of the International Conference on Nonlinear Speech Processing (NOLISP) 2009*, ed. by J. Sole-Casals, V. Zaiats. Lecture Notes on Computer Science (LNCS), vol. 5933/2010 (Springer, Vic, 2010), pp. 68–75

M. Wöllmer, M. Kaiser, F. Eyben, B. Schuller, G. Rigoll, LSTM-Modeling of Continuous Emotions in an Audiovisual Affect Recognition Framework. Image Vis. Comput. (IMAVIS) **31**(2), 153–163. Special Issue on Affect Analysis in Continuous Input (2013)

S. Wu, T.H. Falk, W.-Y. Chan, Automatic speech emotion recognition using modulation spectral features. Speech Commun. **53**(5), 768–785 (2011). doi:10.1016/j.specom.2010.08.013. ISSN 0167-6393 (Perceptual and Statistical Audition)

Q. Yan, S. Vaseghi, E. Zavarehei, B. Milner, J. Darch, P. White, I. Andrianakis, Formant-tracking linear prediction model using hmms and kalman filters for noisy speech processing. Comput. Speech Lang. **21**(3), 543–561 (2007). doi:10.1016/j.csl.2006.11.001

S. Young, G. Evermann, M. Gales, T. Hain, D. Kershaw, X. Liu, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, P. Woodland, *The HTK Book*, Cambridge University Engineering Department, for HTK version 3.4 edition (2006)

E. Yumoto, W.J. Gould, Harmonics-to-noise ratio as an index of the degree of hoarseness. J. Acoust. Soc. Am. (JASA) **71**(6), 1544–1549 (1981)

G. Zhou, J.H.L. Hansen, J.F. Kaiser, Nonlinear feature based classification of speech under stress. IEEE Trans. Speech Audio Process. **9**(3), 201–216 (2001). doi:10.1109/89.905995

X. Zuo, P. Fung, A cross gender and cross lingual study of stress recognition in speech without linguistic features, in *Proceedings of the 17th ICPhS* (Hong Kong, China, 2011)

E. Zwicker, Subdivision of the audible frequency range into critical bands. J. Acoust. Soc. Am. (JASA) **33**(2), 248–248 (1961)

E. Zwicker, Masking and psychological excitation as consequences of ear's frequency analysis, in *Frequency Analysis and Periodicity Detection in Hearing*, ed. by R. Plomp, G.F. Smoorenburg (Sijthoff, Leyden, 1970)

E. Zwicker, E. Terhardt, Analytical expressions for critical-band rate and critical bandwidth as a function of frequency. J. Acoust. Soc. Am. (JASA) **68**, 1523–1525 (1980)

E. Zwicker, H. Fastl, *Psychoacoustics—Facts and Models*, 2nd edn. (Springer, Berlin, 1999), 417 p. ISBN 978-3540650638