Mid-level representations for audio content analysis

*Slides for this lecture were created by Anssi Klapuri

Sources:

Contents
- Introduction
- Desirable properties of mid-level representations
- STFT spectrogram
- Constant-Q transform
- Sinusoids plus noise
- Perceptually-motivated representations

1 Introduction
- The concept of mid-level data representations is useful in characterizing signal analysis systems
- The analysis process can be viewed as a sequence of representations from an acoustic signal ("low level") towards the analysis result ("high")
- Usually intermediate abstraction levels are needed between the two since high-level information is not readily visible in the raw acoustic signal
- An appropriate mid-level representation functions as an "interface" for further analysis and facilitates the design of efficient algorithms

Desirable properties of mid-level representations
- It is natural to ask if a certain mid-level representation is better than others in a given task.
- Ellis and Rosenthal list several desirable qualities for a mid-level representation:
  - Component reduction: the number of objects in the representation is smaller and the meaningfulness of each is higher compared to the individual samples of the input signal
  - The sound should be decomposed into sufficiently fine-grained elements so as to support sound source separation by grouping the elements to their sound sources.
  - Invertibility: the original sound can be resynthesized from its representation in a perceptually accurate way,
  - Psychoacoustic plausibility of the representation.

Categorization of some mid-level representations
- Ellis and Rosenthal classify representations according to three conceptual axes
  - choice between fixed and variable bandwidth of the initial frequency analysis
  - discreteness: is representation structured as meaningful chunks?
  - dimensionality of the transform (some possess extra dimensions)
- Figure: [Ellis-Rosenthal]
2 Complex-valued STFT spectrogram

- STFT = short-time Fourier transform
- Time-domain signal $x(n)$ is transformed into time-frequency domain by employing discrete Fourier transform (DFT) in successive time frames
  - complex spectra: all information is preserved
  - amount of data remains the same
- To some extent fulfills the criteria of
  - supporting sound source separation (sources overlap less in time-frequency than in time domain)
  - invertibility: the original sound can be perfectly reconstructed
  - psychoacoustic plausibility: frequency analysis (albeit in a different form) happens in the auditory system too

STFT spectrogram

- Example signal (music)
  - top: time-domain (zoomed in)
  - middle: time-domain
  - bottom: STFT spectrogram (magnitudes)

Spectrum estimation

- Spectrum of audio signals is typically estimated in short consecutive segments, frames
- Why?
  - the Fourier transform models the signal with stationary sinusoids (constant spectrum)
  - real audio signals are not stationary but vary through time
  - framewise processing assumes the signal is time-invariant in short enough frames
- For audio signals, the frame length typically varies between 10ms – 100ms, depending on the application
  - for speech signals often 25ms
- Transient-like sounds are difficult to represent and process in the frequency domain
  - time blurring (but let’s see constant-Q transform later...)

Windowing

- **Windowing** is essential in frame-wise processing
  - weight the signal with a window function $w(k)$ prior to transform
  - as a rule of thumb, windowing is always needed: one cannot just take a short part of a signal without windowing

signal in frame $m$: $x_m(n), n = 0, ..., N - 1$

windowed signal: $x_m(n)w(n)$

short-time spectrum:

$$X_m(k) = \sum_{n=0}^{N-1} x_m(n)w(n)W^{nk}$$
Windowing

- Example: spectrum of a sinusoid with/without windowing
  1. No windowing (=rectangular window), sinusoid at a spectral bin
  2. No windowing, random off-bin frequency \(\rightarrow\) spectral blurring!
  3. Hanning window, sinusoid at a spectral bin
  4. Hanning window, random off-bin frequency \(\rightarrow\) ok

- There are different types of windows, but most important is not to forget windowing altogether

Windowing in framewise processing

- Figure: Hanning windows (Hamming works too)
  - adjacent windows sum to unity when frames overlap 50% 
  \(\rightarrow\) all parts of the signal get an equal weight
  - In each frame, the signal is weighted with the window function and short-time discrete Fourier transform is calculated
  - This yields a spectrogram
    - complex spectrum in each frame over time

Windowing in analysis-synthesis systems

- Sine window is useful in analysis-synthesis systems (see Figure)
- Windowing is done again in resynthesis to avoid artefacts at frame boundaries in the case that the signal is manipulated in the \(f\)-domain
  - Figure below: 50% frame overlap leads to perfect reconstruction if nothing is done at subbands

Reconstructing the time domain signal: overlap-add technique

- Reconstructing a signal from its spectrogram:
  1. inverse Fourier transform the spectrum of each frame back to time domain
  2. apply windowing in each frame (e.g. sine or Hanning window)
  3. successive frames are positioned to overlap 50% or more, and summed sample-by-sample
3 Constant-Q transform (CQT)

- Time-frequency representation where the frequency bins are uniformly distributed in log-frequency and their Q-factors (ratios of center frequencies to bandwidths) are all equal.
- In effect, that means that the frequency resolution is better for low frequencies and the time resolution is better for high frequencies.
- Musically and perceptually motivated:
  - Frequency resolution of the inner ear is approx. constant Q above 500 Hz.
  - In music (equal temperament), note frequencies obey \( F_k = 440\text{Hz} \times 2^{k/12} \).

\[ X_{\text{CQT}}(k,n) = \sum_{m=0}^{N-1} x(m) g_k(m-n) e^{-j2\pi mkf/s} \]

where \( N \) is the length of the input signal, \( g_k(m) \) is a zero-centered window function that picks one time frame of the signal at point \( n \) and \( f_s \) sample rate.

- Compare CQT with short-time Fourier transform (STFT):
  \[ X_{\text{STFT}}(k,n) = \sum_{m=0}^{N-1} x(m) h(m-n) e^{-j2\pi mkf/s} \]

where now the window function \( h(m) \) is the same for all frequency bins.

- In CQT, to achieve constant Q-factors, the support of the window (time length of significant non-zero values) is inversely proportional to \( f_k \).
- In CQT, the center frequencies are geometrically spaced: \( f_k = f_0 \times 2^{k/B} \) where \( B \) determines the number of bins per octave and \( f_0 \) is the lowest bin.
- In DFT, the center frequencies are linearly spaced: \( f_k = k\text{resolution} \).

CQT is essentially a wavelet transform, but with rather high frequency resolution (typically 10–100 bins/octave)

\[ \rightarrow \text{conventional wavelet transform techniques cannot be used} \]

Matlab Toolbox for CQT and ICQT [Schörkhuber et al. 2013]:
- [http://www.cs.tut.fi/sgn/arg/CQT/]
- Efficient computation achieved by:
  - FFT of the entire input signal
  - apply one CQT frequency-bin wide "bandpass" on the (huge) spectrum
  - move subband around zero
  - inverse-FFT transform the narrowband spectrum to get CQT coefficients over time for that bin.
Due to the way that CQT is computed by the toolbox, the frequency-domain response of an individual frequency bin can be controlled perfectly (no sidelobes), but the effective *time-domain* window has sidelobes (that extend over the entire signal).

Figure: the response of one time-frequency element as a function of frequency (left) and as a function of time (right)

**Toolbox for computing CQT**

- Due to the way that CQT is computed by the toolbox, the frequency-domain response of an individual frequency bin can be controlled perfectly (no sidelobes), but the effective *time-domain* window has sidelobes (that extend over the entire signal).

- Figure: the response of one time-frequency element as a function of frequency (left) and as a function of time (right)

**STFT spectrogram for the same signal**

(either high or low frequencies blur)

**STFT spectrogram for the same signal**

(either high or low frequencies blur)
Representations 22

Reasons why CQT has not more widely replaced FFT in audio signal processing:
1. CQT is computationally more intensive than DFT spectrogram
2. CQT produces a data structure that is more difficult to handle than the time-frequency matrix (spectrogram)

Drawbacks of CQT

Example application: Pitch shifting

The toolbox includes PITCHSHIFT.m to implement that

Pitch shifting is a natural operation in CQT domain:
1. CQT
2. translate CQT coefficients ↓ or ↑ in frequency
3. retain phase coherence
4. inverse CQT

Examples:
- original
- 6 semitones
- +6 semitones

Transients at high freqs. are retained due to short frame
Time stretching can be done by pitch shifting + resampling

4 Sinusoids plus noise model

(Recap from SGN-14006)

Signal model
\[ x(t) = \sum_{n=1}^{N} a_n(t) \cos[2\pi f_n(t) t + \varphi_n(t)] + r(t) \]

\[ \Rightarrow \] signal \( x(t) \) is represented with \( N \) sinusoids (freq, amplitude, phase) and noise residual \( r(t) \)

Additive synthesis
- according to Fourier theorem, any signal can be represented as a sum of sinusoids
- makes sense only for periodic signals, for which the amount of sinusoids needed is small
- non-deterministic part would require a large number of sinusoids
  \[ \Rightarrow \] use stochastic modeling
Sinusoids+noise model
Analysis

- Block diagram [Virtanen 2001]
  1. detect sinusoids in framewise spectra
  2. estimate sinusoid parameters and resynthesize
  3. subtract sinusoids from original signal
  4. model the noise residual
- We get
  - sinusoid parameters
  - noise level at different subbands
**Sinusoids+noise model**

**Detecting and estimating sinusoids**

- **Block diagram:** [Virtanen01]
- Spectral peaks are interpreted as sinusoids
  1. "peak": local maximum in magnitude spectrum
  2. peak frequency, amplitude, and phase can be picked from the complex spectrum

- Tracking the peaks detected in successive frames
  → gives parameters of a time-varying sinusoid
  → "sinusoidal trajectory"

**Synthesis of sinusoids**

- **Additive synthesis**
  \[ s(t) = \sum_{n=1}^{N} a_n(t) \cos[2\pi f_n(t) t + \varphi_n(t)] \]

- Often tracking the peaks is not necessary, but
  - synthesize sinusoids in each frame separately, keep the parameters fixed in one frame
  - window the obtained signal with Hann window
  - overlap-add

**Tracking the peaks**

- **If needed**, spectral peaks in successive frames can be associated and joined into time-varying sinusoids
  → frequency, amplitude, and phases joined into curves

- **Figure:** peak tracking algorithm [Virtanen2001]
  - based e.g. on the track derivatives; try to form a smooth track
  - kill: if no continuation found, end the sinusoid
  - birth: if spectral peak is not a continuation for an existing sinusoid, create a new one

**Synthesis, subtraction from original**

- Synthesized sinusoids vs. the original signal (upper panel)
- Residual obtained from subtraction (lower panel)
Sinusoids+noise model

Modeling the noise residual

- Residual is obtained by subtracting synthesized sinusoids from the original signal in the time-domain
- Residual signal is analyzed frame-by-frame
  - calculate spectrum $R_t(f)$ in frame $t$
  - subdivide the spectrum into 25 perceptual subbands (Bark scale)
  - calculate short-time energy at each band $b, b = 1, 2, \ldots, 25$

$$E_t(b) = \sum_{f \in b} |R_t(f)|^2$$

Sinusoids+noise model

Noise synthesis from parameters

- Noise residual is represented parametrically
  - in each frame, store only the short-time energies within Bark bands, $E_t(b)$
  - this modeling can be done, because the auditory system is not sensitive to energy changes within one Bark band in the case of noise

$$R_t(f) = \sqrt{E_t}$$

Synthesis
1. generate magnitude spectrum, where the energy within each Bark band is shared uniformly within the band
2. generate random phases
3. inverse Fourier transform to time-domain
4. windowing with Hann window
5. overlap-add

Sinusoids+noise model

Properties

- Audio examples:
- Sinusoids+noise model has several nice properties
  - satisfies the "component reduction" property (see slide 3)
  - invertibility: synthesized signal has reasonable quality (transient sounds are problematic)
  - the model is generic: any sound can be processed
  - straightforward to compute (especially if peak tracking is skipped)
  - manipulation such as time stretching and pitch shifting is easy
- The representation also supports sounds source separation to some extent, see next slides

Intelligent component grouping

- Auditory organization in humans has been found to depend on certain acoustic cues
- Two components may be grouped, i.e., associated to a common sound source by
  1. Spectral proximity (time, frequency)
  2. Harmonic concordance
     - frequencies components in integral ratios
       → these components are deduced to be produced by a common source
  3. Synchronous changes of the components
     - common onset / offset
     - common AM / FM modulation
     - equidirectional movement in the spectrum
  4. Spatial proximity (angle of arrival)
- Cues may compete and conflict
Example: grouping sinusoids

- Sinusoidal model reveals the cues better than the time-domain signal

Sound separation
Example: grouping implemented

- Estimate *perceptual distance* between each two spectral components
  - sinusoids are classified into groups

5 Perceptually-motivated representations

Peripheral hearing
1. Frequency selectivity of the inner ear
   - Bank of linear bandpass filters
     - "auditory channels"
2. Mechanical-to-neural transduction
   - Compression, rectification, lowpass filtering
   - Detailed models exist, too

In brain, for pitch processing:
3. Periodicity analysis within channels
4. Combination across channels
   - Between-channel phase differences do not affect (yet)
### Perceptually-motivated representations

- The signal traveling in the auditory nerve fibers from the auditory periphery to the brain can be viewed as a mid-level representation.
- The idea of using the same data representation as the human auditory system is very appealing.
- Auditory periphery is quite accurately known.

### Auditory filterbank

- Band-wise processing is an inherent part of hearing.
- **Figure**: Frequency responses (top) and impulse response (bottom) of a few auditory filters.
  - Bandwidths proportional to center frequency:
    
    
    \[ b_c = 0.108 f_c + 24.7 \text{Hz} \]

### Mechanical-to-neural transduction

#### Simplified model:

- a. Compression (and level adaptation)
- b. Half-wave rectification
- c. Lowpass filtering

**Compression**:

- Memoryless: scale the signal with a factor
  \[ a_c = (\sigma_c)^{\nu-1} \]
  where \( \sigma_c \) is the std of the signal within channel \( c \)
- Spectral flattening (whitening) when \( 0 < \nu < 1 \)

**Half-wave rectification within subbands**

\[ \text{input partials + beating partials (freq. intervals btw the input partials)} \]
Performing periodicity analysis within critical bands produces a three-dimensional volume $r_c(n, r)$ for channel $c$ at time $n$ and lag $r$. The extent to which harmonic $h$ is mapped to the position of the fundamental increases as a function of $h$.

**Amplitude-modulated noise**

**Correlogram**

Input signal was a trumpet sound with F0 260 Hz (period 3.8 ms). The figure below illustrates the correlogram:

- Left: the 3D correlogram (power spectrogram)
- Middle: zero-lag face of the correlogram
- Right: one time slice of the volume, from which summary ACF can be obtained by summing over frequency.

**Autocorrelation within channels**

Summary autocorrelation: combining across channels

$$s(n, r) = \sum_{i=0}^{n} z(n-i)z(n-i-r)w(i)$$

$$r_c(n, r) = \sum_{i=0}^{n} z_c(n-i)z_c(n-i-r)w_c(i)$$