M2 ATIAM

Audio Features
Automatic Chord Estimation

Geoffroy Peeters

contact: geoffroy.peeters@telecom-paris.fr
Télécom-Paris, IP-Paris, France

Music Information Retrieval → Applications

Extracting/ Estimating

- **Automatic Music Transcription**
  - onset-detection
  - pitch, multi-pitch, dominant melody
  - chords/guitar tab, key
  - rhythm: tempo, meter, beat, downbeat
  - musical instrument
  - lyrics
- **Auto-tagging**
  - genre, mood, context, ...
- **Music Recommandation**
  - similarity, cover-detection

Processing

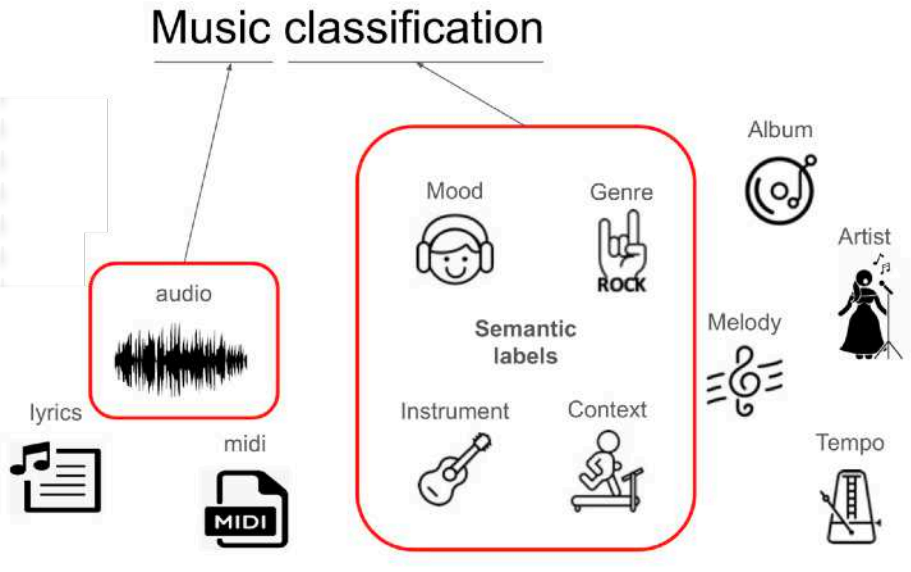
- **Source separation, enhancement**

Generating

- **Audio/ Music generation, transfer**

ISMIR

Spotify DEEZER

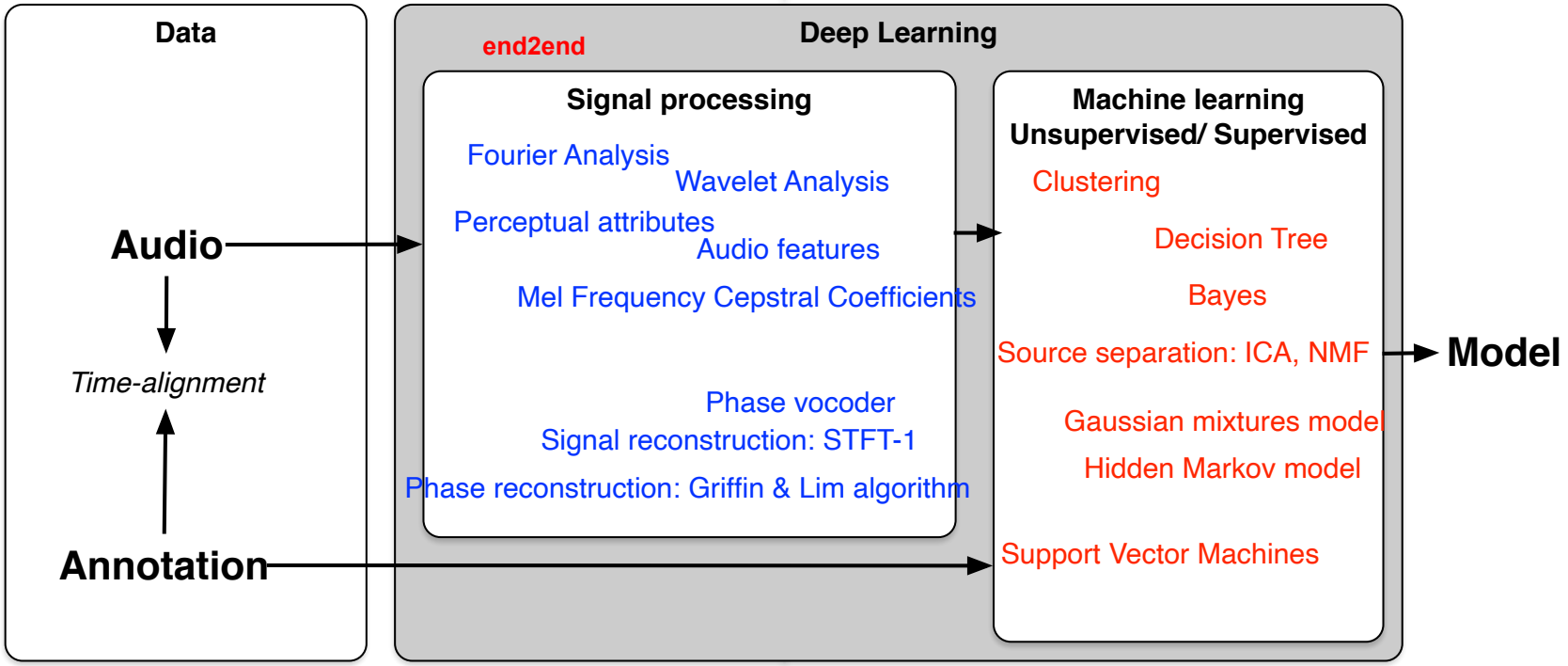


Celma, Oscar, 2010 "Music recommendation", *Music recommendation and discovery, Springer.*

Various categories of audio content
Applications

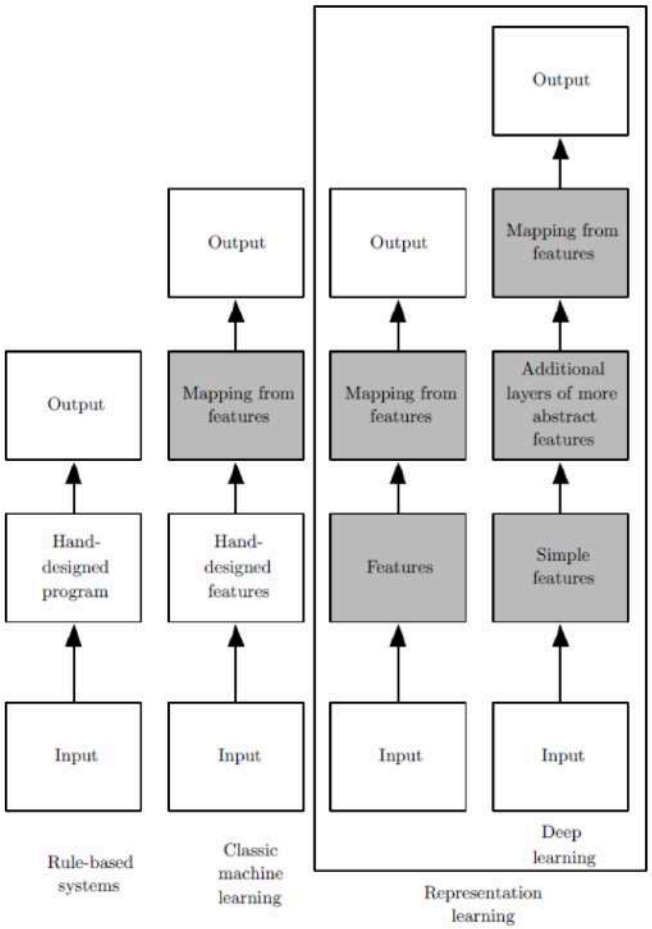
	Speech	Music	Envirronmental Sounds
Description	Speech to text (ASR) Speaker recognition Speaker diarization	Audio ID (Shazam) Auto-tagging Recommendation Content-description (pitch, chord, tempo) Lyrics alignment/ recognition	Acoustic scene classificaton Sound event localization, detection
Transformation	Speech separation, enhancement Speech coding Speech transformation	Singing separation (Karaoke) Unmixing Style transfer	
Generation	Text to speech	Sound generation Music generation (OpenAI)	Sound texture synthesis
		MIR http://ismir.net	DCASE http://dcase.community

Traditional approaches for audio



What is deep learning ?

Deep learning: learning hierarchical representations



From Deep Learning by Ian Goodfellow and Yoshua Bengio and Aaron Courville

Traditional machine-learning approach
(1) Audio features

Audio features

- Various **forms**:
  - **scalar**: spectral centroid, spectral spread, fundamental frequency, spectral roll-off, spectral flux, zero-crossing rate, RMS, ...
  - **vector**: Mel Frequency Cepstral Coefficients, coefficients LPC, coefficients PLP, ...

- Various **time validity**:
  - represent one **frame** of the audio signal → "instantaneous" feature
  - represent the content of a **set of local frames** →texture windows
  - represent **globally** the audio signal

- Highlight different facets of the audio **content**:
  - **timbre** content: Mel Frequency Cepstral Coefficients, LPC coefficients, PLP coefficients, ...
  - **harmonic** content: Pitch Class Profiles/ Chroma, ...
  - **noise** content: Spectral Flatness Measure, ...
  - **rhythmic** content: ...

[G. Peeters. *A large set of audio features for sound description (similarity and classification) in the cuidado project*. Cuidado project report, Ircam, 2004.]

Audio features examples

Zero-crossing rate (zcr)

- Measures the number of times the audio waveform cross the zero-axis
  - $$zcr = \frac{1}{N}\sum_{n=1}^{N} |sign(x_n) - sign(x_{n-1})|$$
- Usage: allows to distinguish
  - harmonic sounds → low zcr
  - noise sounds→ high zcr

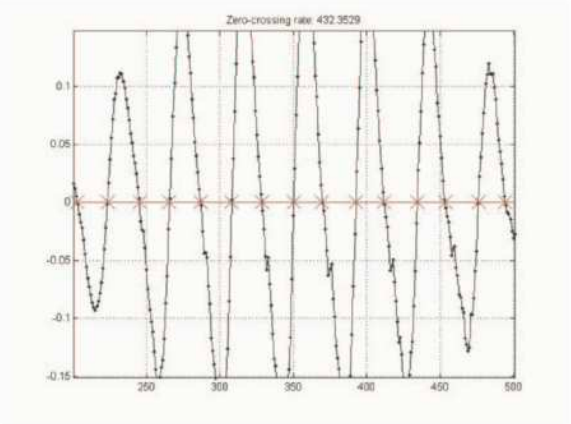


Figure 12 Zero-crossing rate (=432) during voiced speech region

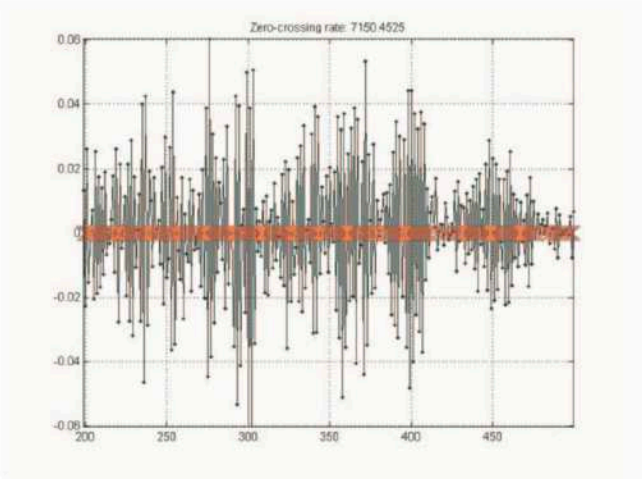


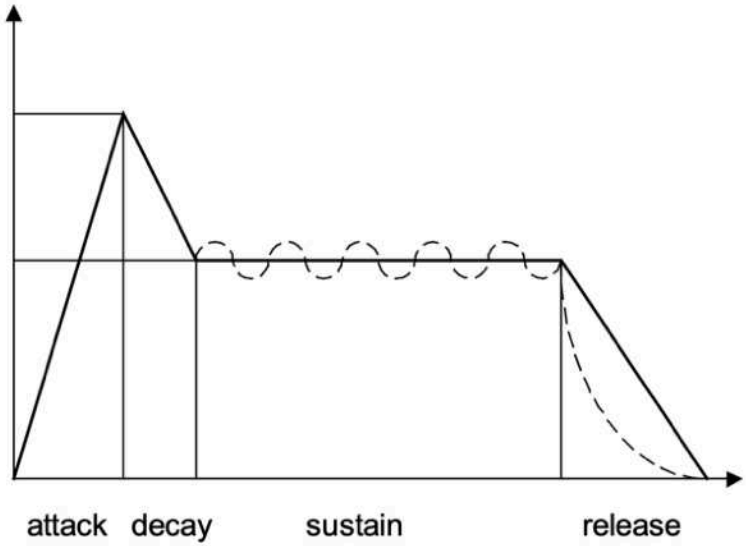
Figure 13 Zero-crossing rate (=7150) during unvoiced speech region

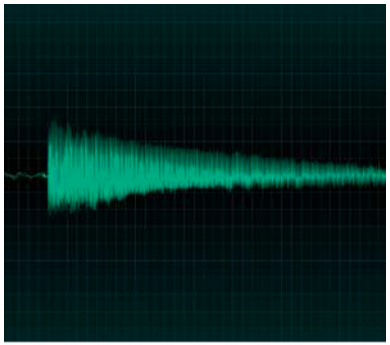
Audio features examples

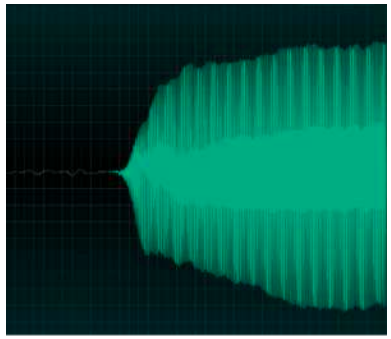
ADSR (Attack, Decay, Sustain, Release) temporal enveloppe

- Model of the temporal evolution (enveloppe) of the energy of a musical note

- Usage: allows to distinguish
  - fast attacks (percussive sounds) /slow attacks
  - fast decrease(non-sustained sounds) / slow decrease (sustained sounds)





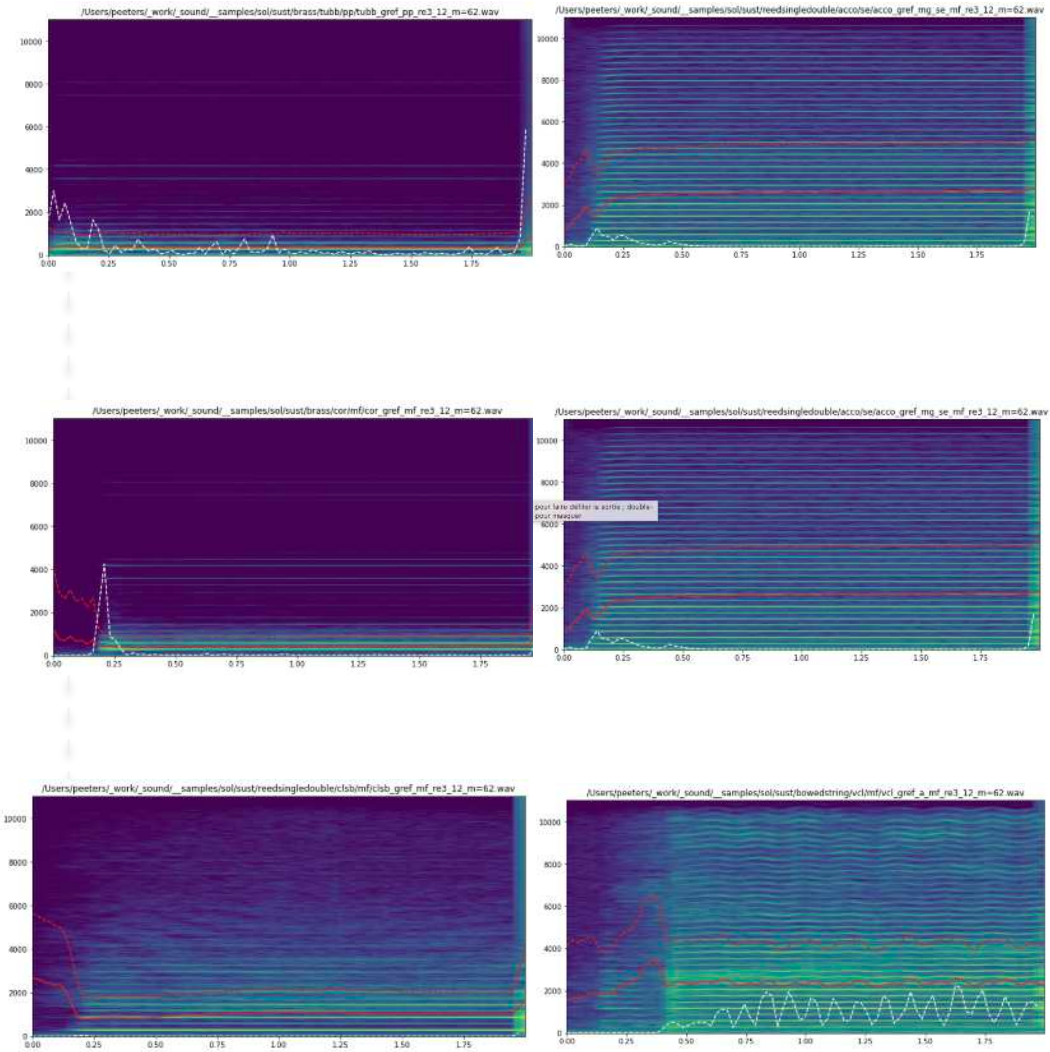


Audio features examples

Spectral shape description

- **Spectral centroid**
  - $$cs = \frac{\sum_k f_k A_k}{\sum_k A_k}$$
  - allows to distinguish between "dull" and "bright" sounds
- **Spectral spread**
  - $$es = \sqrt{\frac{\sum_k (f_k - cs)^2 A_k}{\sum_k A_k}}$$
  - allows to distinguish between "poor" and "rich" sounds
- **Spectral flux**
  - Measure the temporal variation of the spectrum

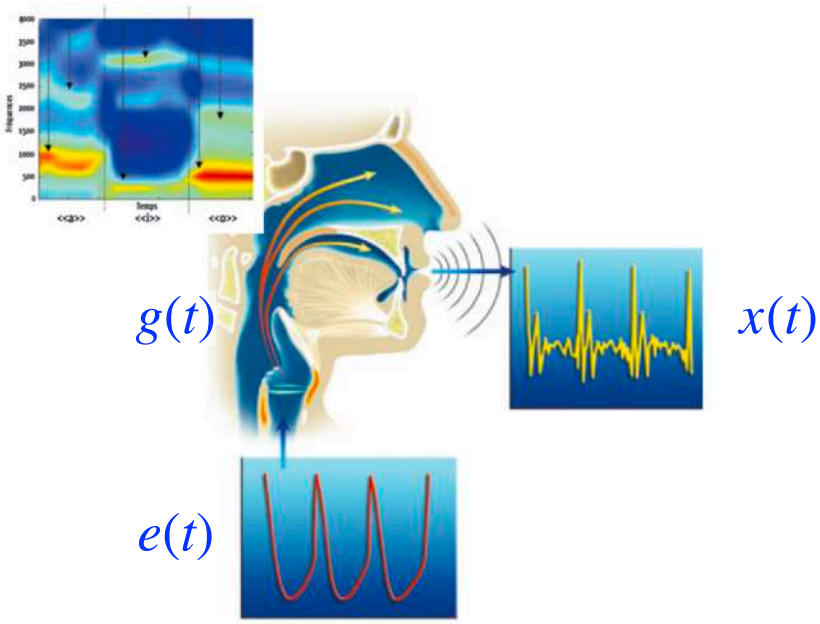
    $$fs = \sum_k (A_k(t) - A_k(t-1))^2$$
  - allows to distinguish between "poor" and "rich" sounds

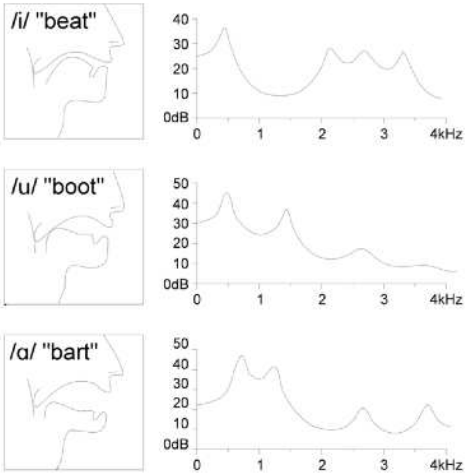


Source/Filter model

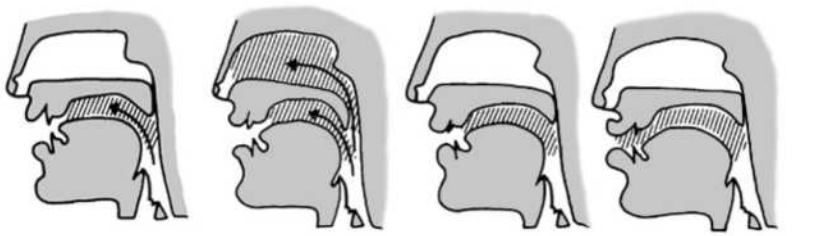
Model:

- represent the signal $x(t)$ as the results of a periodic pulse signal convolved with a resonant filters
- Example:
  - speech (voiced part)
    - vocal folds $e(t)$
    - mouth (resonance), nose (anti-resonance) $g(t)$
  - many musical instruments (trumpet)





source : Mike Brookes



source : outilsrecherche.over-blog.com

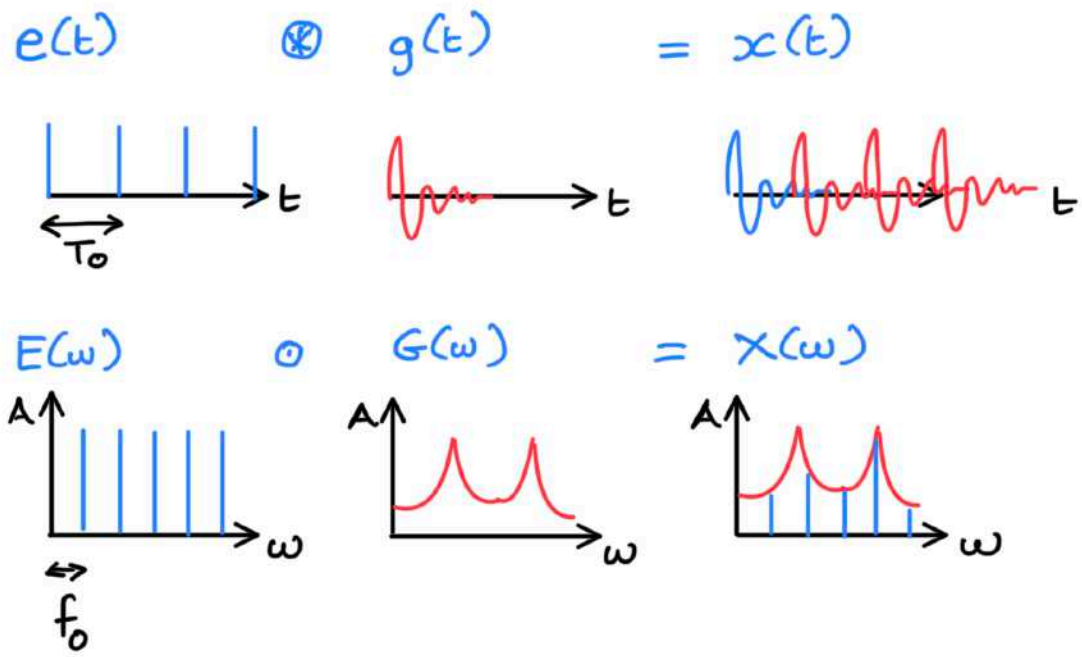
Source/Filter model

In time:

$$x(t) = e(t) \circledast g(t)$$

In frequency:

$$X(\omega) = E(\omega) \cdot G(\omega)$$



Complex cepstrum

– **Goal**

- describe the shape of the spectrum (the timbre) of a signal using a reduced set of coefficients

– **Complex cepstrum** $c(\tau)$

$$c(\tau) = TF^{-1}\left[\log(X(\omega))\right]$$
$$= \frac{1}{2\pi}\int_{\omega} \log[X(\omega)] \cdot e^{j\omega\tau} d\omega$$

- τ is named "**quefrency**" (=frequency in reverse order)
- $x(t) \xrightarrow{TF} X(\omega) \xrightarrow{\log} \log(X(\omega)) \xrightarrow{TF^{-1}} c(\tau)$

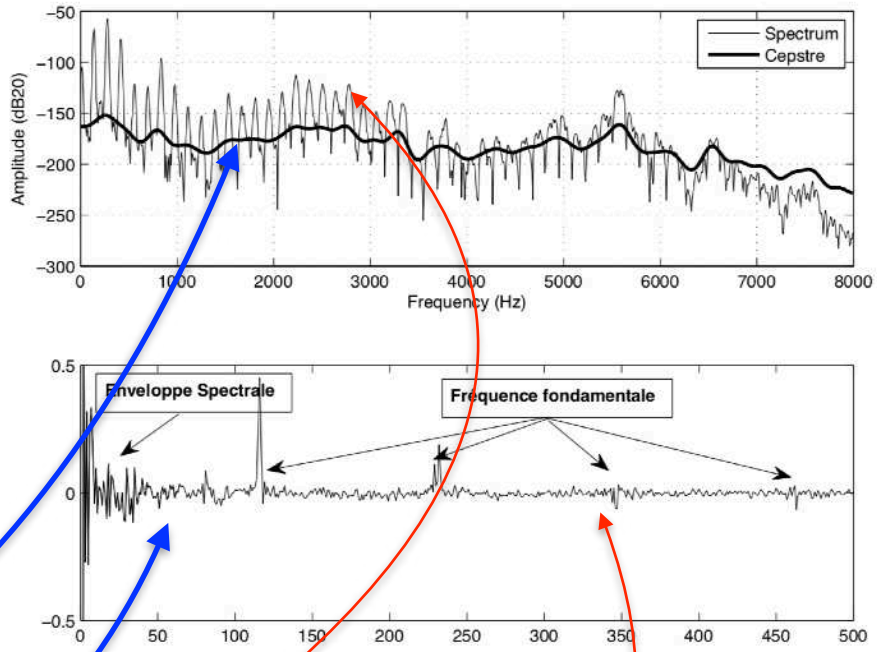
Audio features examples
Mel Frequency Cepstral Coefficients (2)

Source/filter model

- Source $e(t)$: periodic signal
- Filter $g(t)$: resonant/ anti-resonant filter

$$x(t) = e(t) \circledast g(t)$$

$$\xrightarrow{TF} X(\omega) = E(\omega) \cdot G(\omega)$$



$$\xrightarrow{\log} \log(X(\omega)) = \underbrace{\log[G(\omega)]}_{\text{slow variations over } \omega} + \underbrace{\log[E(\omega)]}_{\text{fast variations over } \omega}$$

$$\xrightarrow{TF^{-1}} TF^{-1}\left[\log(X(\omega))\right] = \underbrace{TF^{-1}\left[\log[G(\omega)]\right]}_{\text{energy at quefrency } \tau<<} + \underbrace{TF^{-1}\left[\log[E(\omega)]\right]}_{\text{energy at quefrency } \tau>>}$$

Real cepstrum

– **Real ?** = cepstrum computed on the real part of the log-spectrum

$$X(\omega) = A(\omega) \cdot e^{j\phi(\omega)}$$
$$\log[X(\omega)] = \log[A(\omega)] + j\phi(\omega)$$
$$\Re\{\log[X(\omega)]\} = \log[A(\omega)]$$

$$\text{real cepstrum} = TF^{-1}\left[\Re\{\log[X(\omega)]\}\right]$$
$$= TF^{-1}\left[\log[A(\omega)]\right]$$
$$c(\tau) = \frac{1}{2\pi}\int_{\omega} \log[A(\omega)] \cdot e^{j\omega\tau} d\omega$$

– The amplitude spectrum $A(\omega)$ is real and symmetric
  - its Fourier Transform reduces to the real part
    – reduces to the projection of $\log[A(\omega)]$ on a set of cosinus → Discrete Cosine Transform (DCT)

Audio features examples
Mel Frequency Cepstral Coefficients (4)

Mel Frequency Cepstral Coefficients (MFCCs)

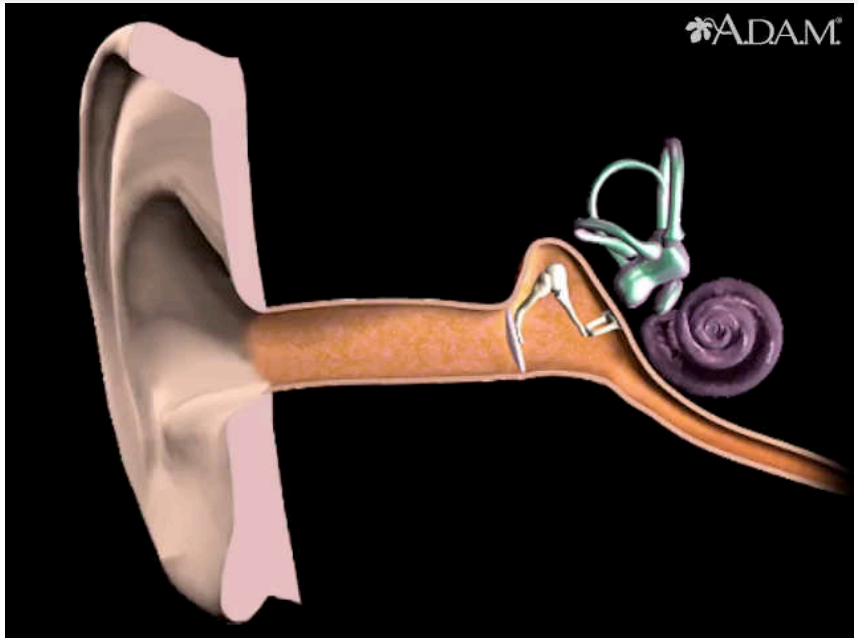
- MFCC ? = real cepstrum computed on the power spectrum $|X(\omega)|^2$ converted to the Mel scale (a perceptual scale)

- **Why perceptual scales ?**
  - Fourier Transform
    - decomposition on a set of sinusoidal components which frequencies are linearly spaced (f_k = 10Hz, 20Hz, 30Hz, ... Hz)
  - Human hearing:
    - decomposition on a set of filters which frequencies are logarithmically spaced (10, 20, 40, 80, ... Hz).
    - highest resolution in low-frequencies, lowest resolution in high frequencies
    - in speech, formants/resonances are closer together in low frequencies
  - MFCCs allows a more compact representation than the real cepstrum
- **How ?**
  - Use of perceptual scales: Mel-scale, Bark-scale, ERB-filters, Gamma-tone filters
- **Usage ?**
  - MFCCs are the most used features in audio: speech, music, environmental sounds recognition, ...

Audio features examples
Mel Frequency Cepstral Coefficients (5)

Human hearing

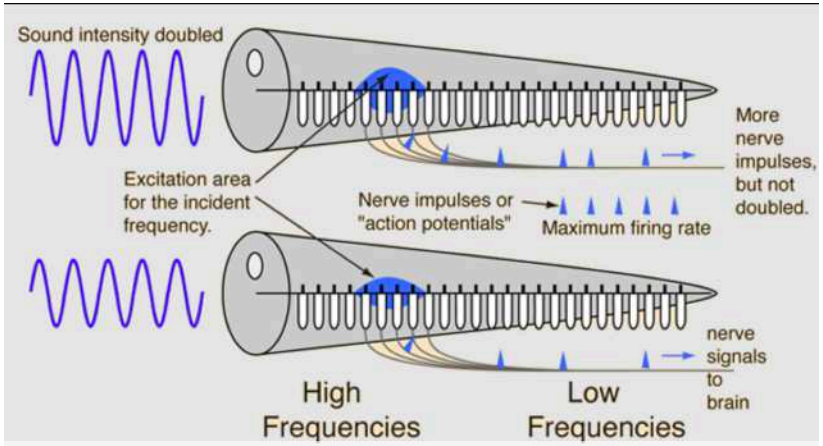
- Cochlea



https://medlineplus.gov/ency/anatomyvideos/000063.htm

- Critical bands
  - perception of two tones at f_1 and f_2
  - perception of a beating-tone at $\frac{f_1+f_2}{2}$

$$\cos f_1 + \cos f_2 = 2\cos\frac{f_1-f_2}{2}\cos\frac{f_1+f_2}{2}$$



source: http://hyperphysics.phy-astr.gsu.edu/hbase/Sound/loud.html

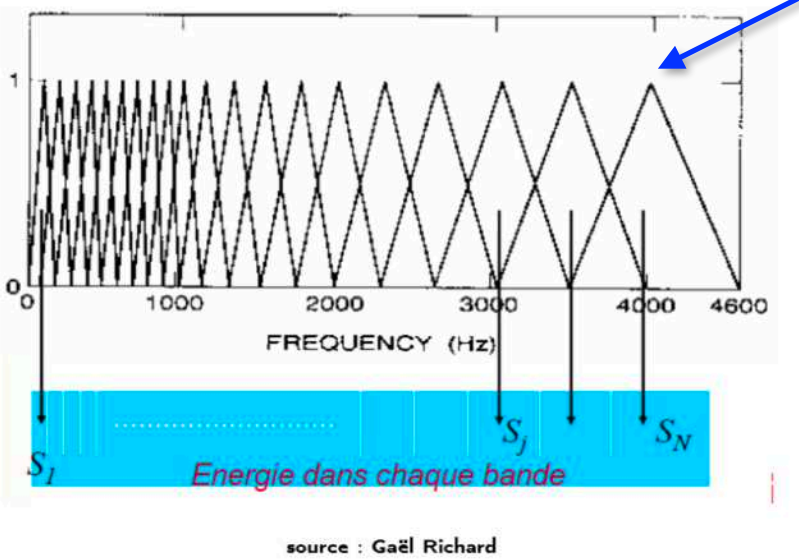
Audio features examples
Mel Frequency Cepstral Coefficients (6)

Mel scale ?

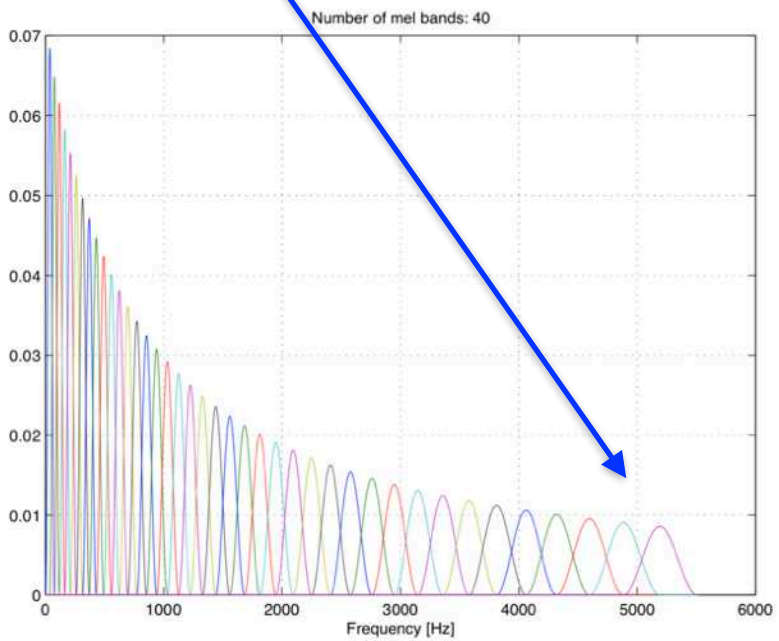
$$mel(f) = \frac{1000}{\ln 2} \ln\left(1 + \frac{f}{1000}\right)$$

- Remark: variations of the constant exist

different shapes for the filter: triangular, hanning, tanh



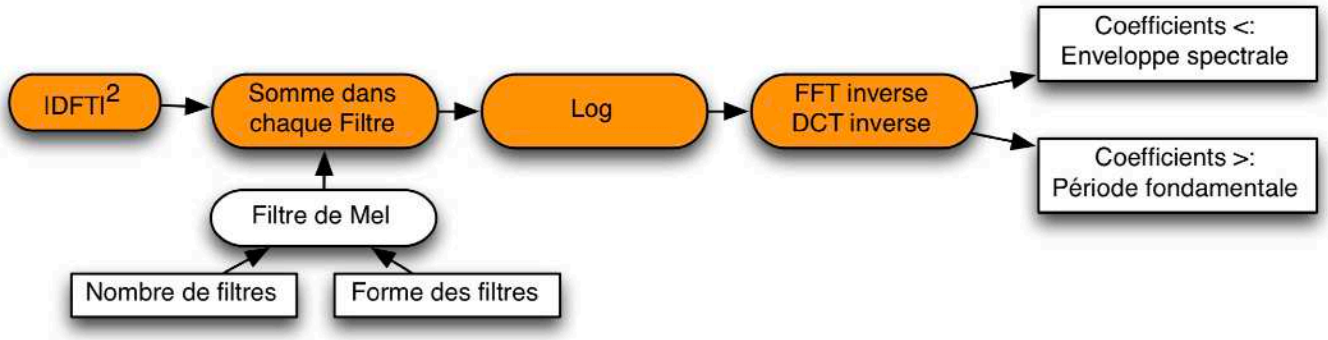
source : Gaël Richard



Fant, Gunnar. (1968). *Analysis and synthesis of speech processes.* In B. Malmberg (Ed.), *Manual of phonetics* (pp. 173-177). Amsterdam: North-Holland.

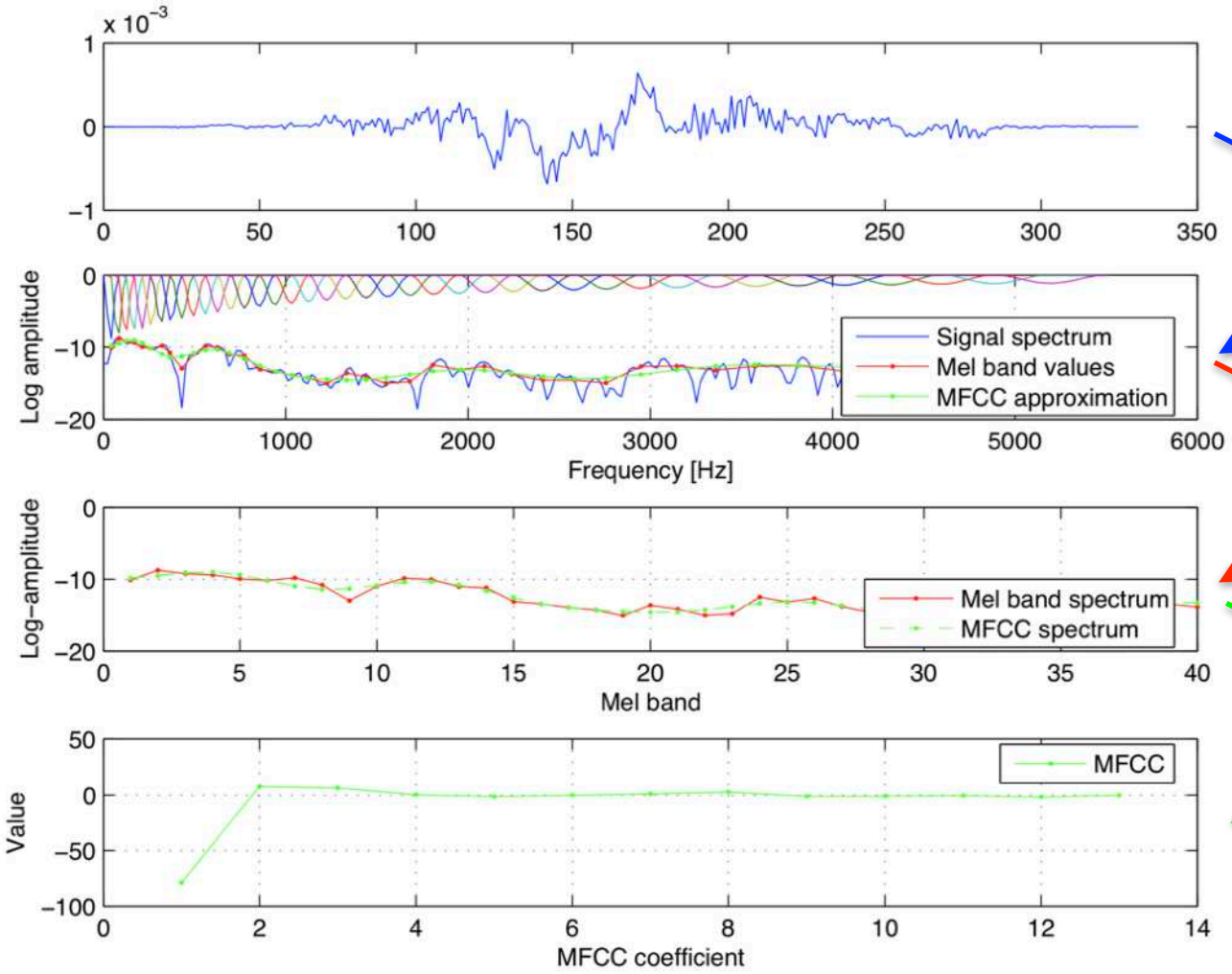
Computation steps for MFCCs

- Compute the power spectrum: $|X(\omega)|^2$
- Compute the Mel filters: $H_b(\omega)$ with $b \in [1,B]$
  - choice of the number of filters B: 40
  - choice of the shape of each filter: triangular, hanning, tanh, ...
- Convert the power spectrum to Mel bands: $S(b) = \sum_{\omega} |X(\omega)|^2 \cdot H_b(\omega)$
- Convert to logarithmic scale: $\log(S(b))$
- Compute the IFFT (or the IDCT): $c(\tau)$
- Select the first coefficients, close to 0 (usually the first 13 coefficients)
  - coefficients close to zero represent the decomposition of the Mel bands content on a set of cosinus with slow variations



Audio features examples
Mel Frequency Cepstral Coefficients (8)

Example of the computation of MFCCs



$|X(\omega)|^2$

$S(b)$

$c(\tau)$

$c(\tau \in [1,13])$

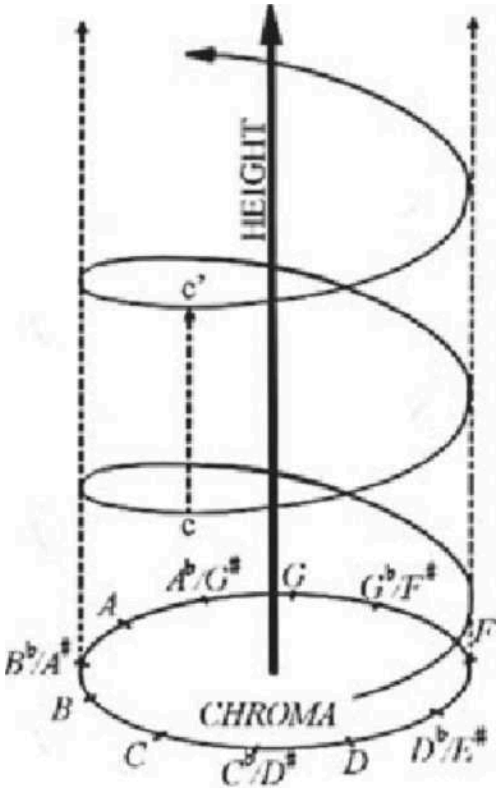
feature of deep learning

first value in order to get low freq

Chroma/ Pitch Class Profile (PCP)

Chroma/ Pitch Class Profile (PCP)

- **Helical model of pitch [Roger Shepard, 1964]**
  - represents the pitch of a note p as a two-dimensional structure:
  - $p = c + o \cdot 12$
    - chroma c (pitch-class)
    - tonal height o (octave number)

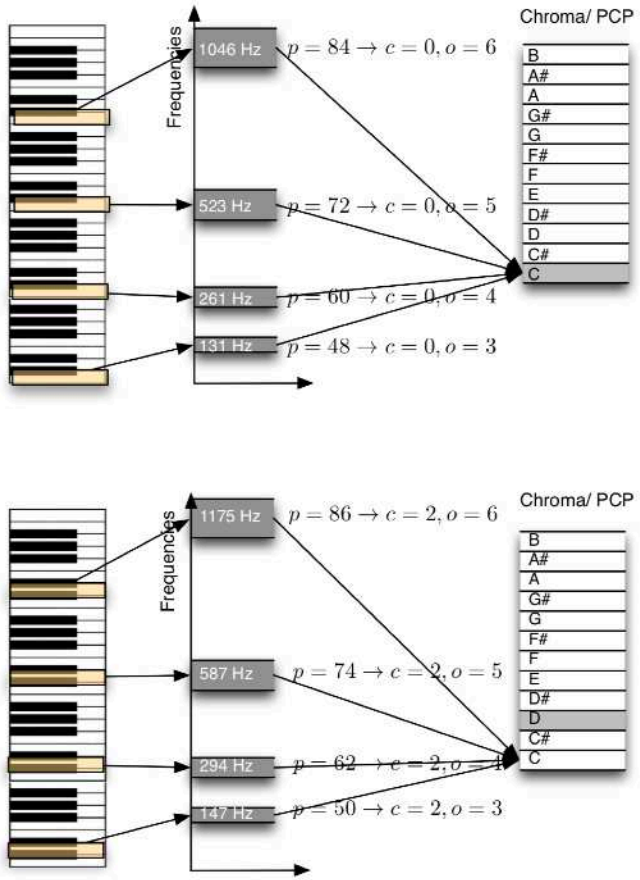


- **Definition: Chroma - Pitch Class Profile (PCP):**
  - represents the harmonic content of the spectrum at time n, $X(k,n)$, as a vector:
    - $C(c,n)$ $c \in [0,12[$

- **Usage:**
  - key estimation,
  - chord estimation,
  - cover detection

Chroma/ Pitch Class Profile (PCP)

- **Chroma computation** $C(c,n)$
  - We sum up the values of the spectrum $X(k,n)$ for all f_k which correspond to a given c
  - Relationship between the frequencies f_k of the DFT and the pitches p (semi-tone pitches in MIDI-scale)
    - $p(f_k) = 12\log_2\left(\frac{f_k}{440}\right) + 69,\ p \in \mathbb{R}^+$
    - $f(p) = 440 \cdot 2^{\frac{p-69}{12}}$



T. Fujishima. Realtime chord recognition of musical sound: a system using common lisp music. In Proc. of ICMC (International Computer Music Conference), pages 464–467, Bejing, China, 1999.

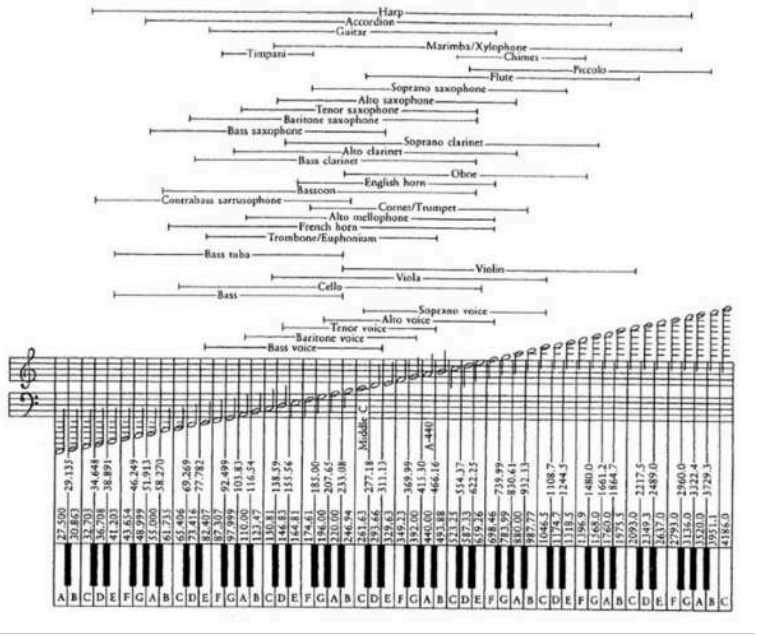
G. H. Wakefield. Mathematical representation of joint time-chroma distributions. In Proc. of SPIE conference on Advanced Signal Processing Algorithms, Architecture and Implementations, pages 637– 645, Denver, Colorado, USA, 1999.

Chroma/ Pitch Class Profile (PCP)

- **Spectral resolution ?**
  - Should allow separating adjacent musical notes
  - We define the width (at -6 dB) as $Bw = \frac{Cw}{L_{sec}}$
  - If $f_{\min}$ (the lowest frequency we consider in the spectrum) is 50 Hz
    - We need to separate $G\#1$ (51.91Hz) from $A1$ (55Hz)

      $$\rightarrow L_{sec} = \frac{Cw}{Bw} = \frac{2.35}{3.0869Hz} = 0.7613s$$
  - If $f_{\min}$ is 100 Hz
    - We need to separate $G\#2$ (103.82Hz) from $A2$ (110Hz)

      $$\rightarrow L_{sec} = \frac{Cw}{Bw} = \frac{2.35}{6.1738Hz} = 0.3806s$$



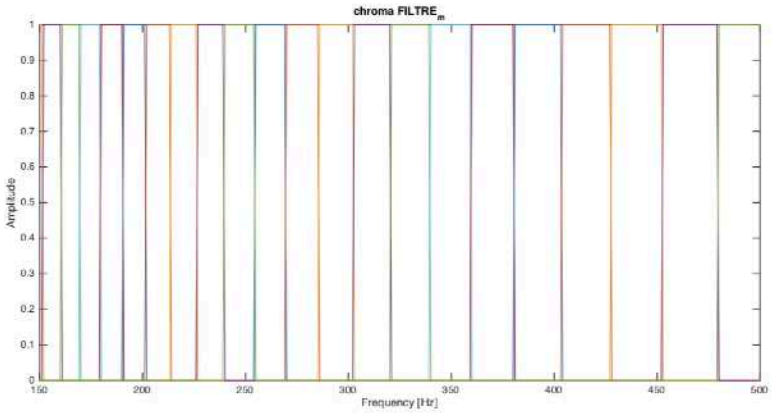
- Two possibilities:
  - Choice L_{sec} as a function of $f_{\min}$
  - Choose $f_{\min}$ as a function of L_{sec}

Chroma/ Pitch Class Profile (PCP)

- **Chroma computation** $C(c,n)$
  - We sum up the values of the spectrum $X(k,n)$ for all f_k which correspond to a given c
  - 1) Hard-mapping
  - 2) Soft-mapping

Chroma/ Pitch Class Profile (PCP)

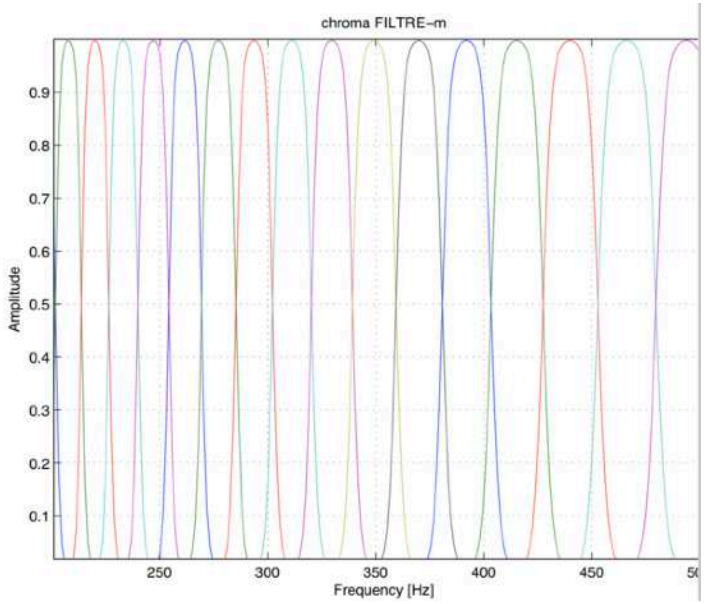
- **1) Hard-mapping ?**
  - A frequency f_k of the DFT only contributes to the closest pitch
  - Example:
    - the energy at $f_k = 452\ Hz\ (p(f_k) = 69.4658)$ only contributes to the pith p=69 (c=10)
    - while $f_k = 453\ Hz\ (p(f_k) = 69.5041)$ to p=70 (c=11).
  - Creation of bank of filters $H_{p'}$ centered on the semi-tone pitches $p' \in \{43,44,\ldots,95\}$:



Chroma/ Pitch Class Profile (PCP)

- **2) Soft-mapping ?**

  - A frequency f_k of the DFT contribues to different chromas with a weight inversely proportional to the distance between $p(f_k)$ and the p the closest
  - Example:
    - the energy at $f_k = 452\ Hz\ (p(f_k) = 69.4658)$ contributes nearly equally to p=69 (c=10) and p=70 (c=11).
  - Creation of bank of filters $H_{p'}$ centered on the semi-tone pitches $p' \in \{43,44,\ldots,95\}$:
    - Each filter is defined by the function
      - $H_{p'}(f_k) = \frac{1}{2}\tanh\left(\pi(1-2x)\right) + \frac{1}{2}$
    - where
      - x = relative distance between the center of the filter p' and the frequencies of the DFT $p(f_k)$
      - $x = R\left|p' - p(f_k)\right|$
    - The filters are evenly distributed and symetrical on the logarithmic scale of semi-tone pitches, non-zero between $p'-1$ and $p'+1$ with a maximum value at p'

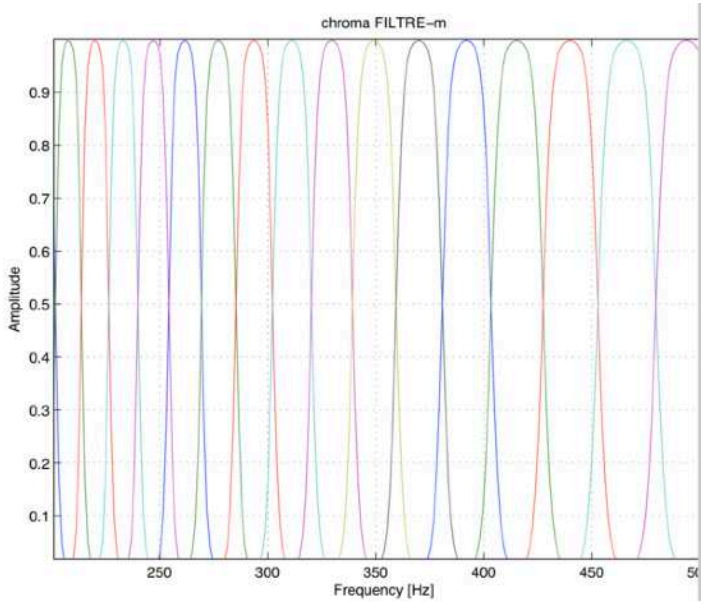


Chroma/ Pitch Class Profile (PCP)

- **2) Soft-mapping (cont.)**
  - The value of the semi-tone pitch spectrum $N(n')$ is given by multiplying the values of the DFT $A(f_k)$ with the bank of filters $H_{n'}$:

$$P(p') = \sum_{f_k} H_{p'}(f_k) A(f_k)$$

  - The mapping between semi-tone pitches n and the pitch-classes (chroma) c is defined by:
    - $c(p) = mod\,(p,12)$
  - The value of the chroma is obtained by summing up the values of equivalent semi-tone pitches
    - $C(c) = \sum_{p' \text{ tel que } c(p')=l} P(n') \quad c \in [0,12[$



Chroma/ Pitch Class Profile (PCP)

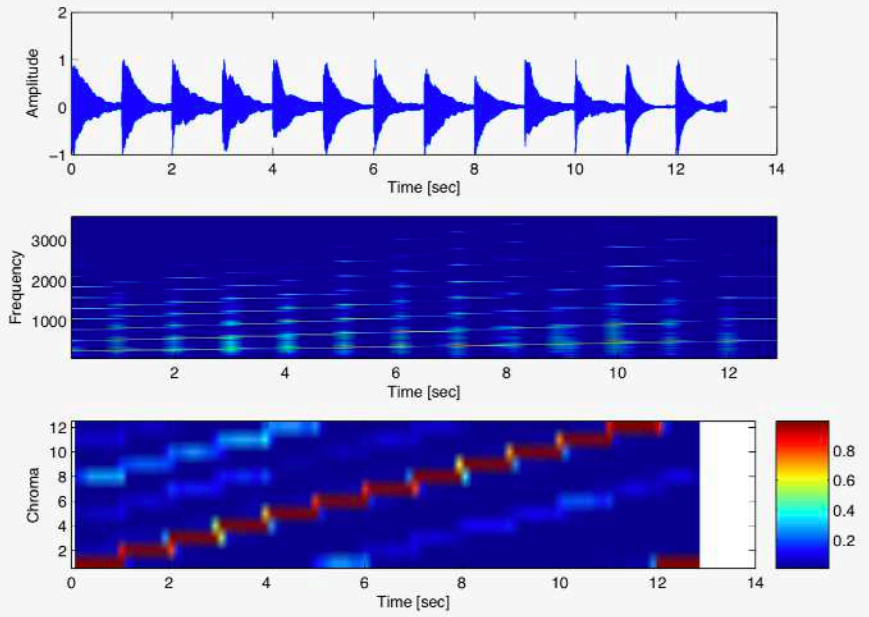
- **Limitations of Chromas - Pitch Class Profile (PCP)**
  - Presence of the upper harmonics of each note
    - In practice, for a given note C we don't have $[1,0,0,0,0,0,0,0,0,0,0,0]$
    - but rather $[a_1 + a_2 + a_4,0,0,0,a_5,0,0,a_4,0,0,0,0]$
    - Influence of the spectral envelope

Pitch	Harmonic	Frequency f_μ	MIDI-scale m_μ	Chroma/PCP p
c3	f_0	130.81	48	1 (=c)
	$2f_0$	261.62	60	1 (=c)
	$3f_0$	392.43	67.01	8.01 ($\simeq$ g)
	$4f_0$	523.25	72	1 (=c)
	$5f_0$	654.06	75.86	4.86 ($\simeq$ e)
	...	...	...	...

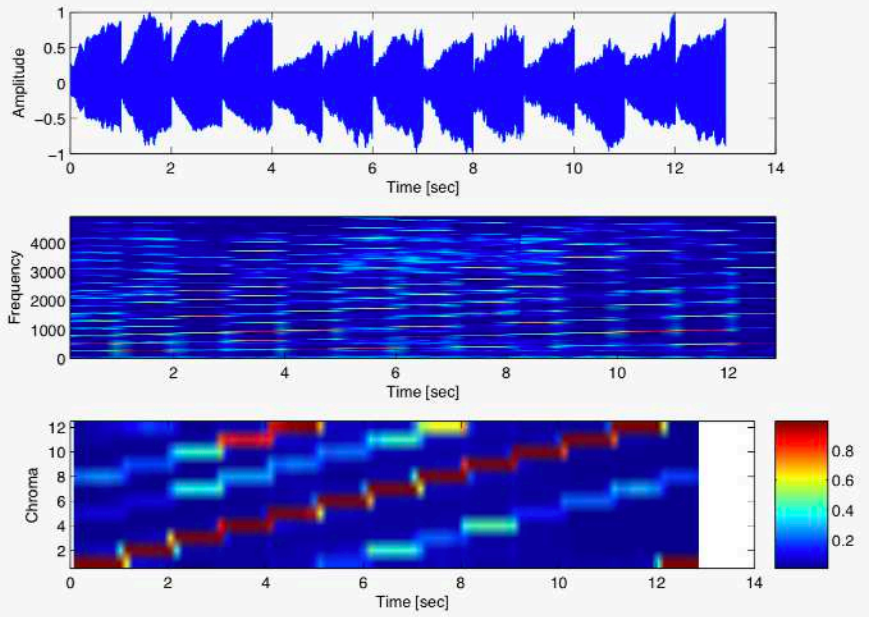
Chroma/ Pitch Class Profile (PCP)

- **Limitations of Chromas - Pitch Class Profile (PCP)**

Exemple piano

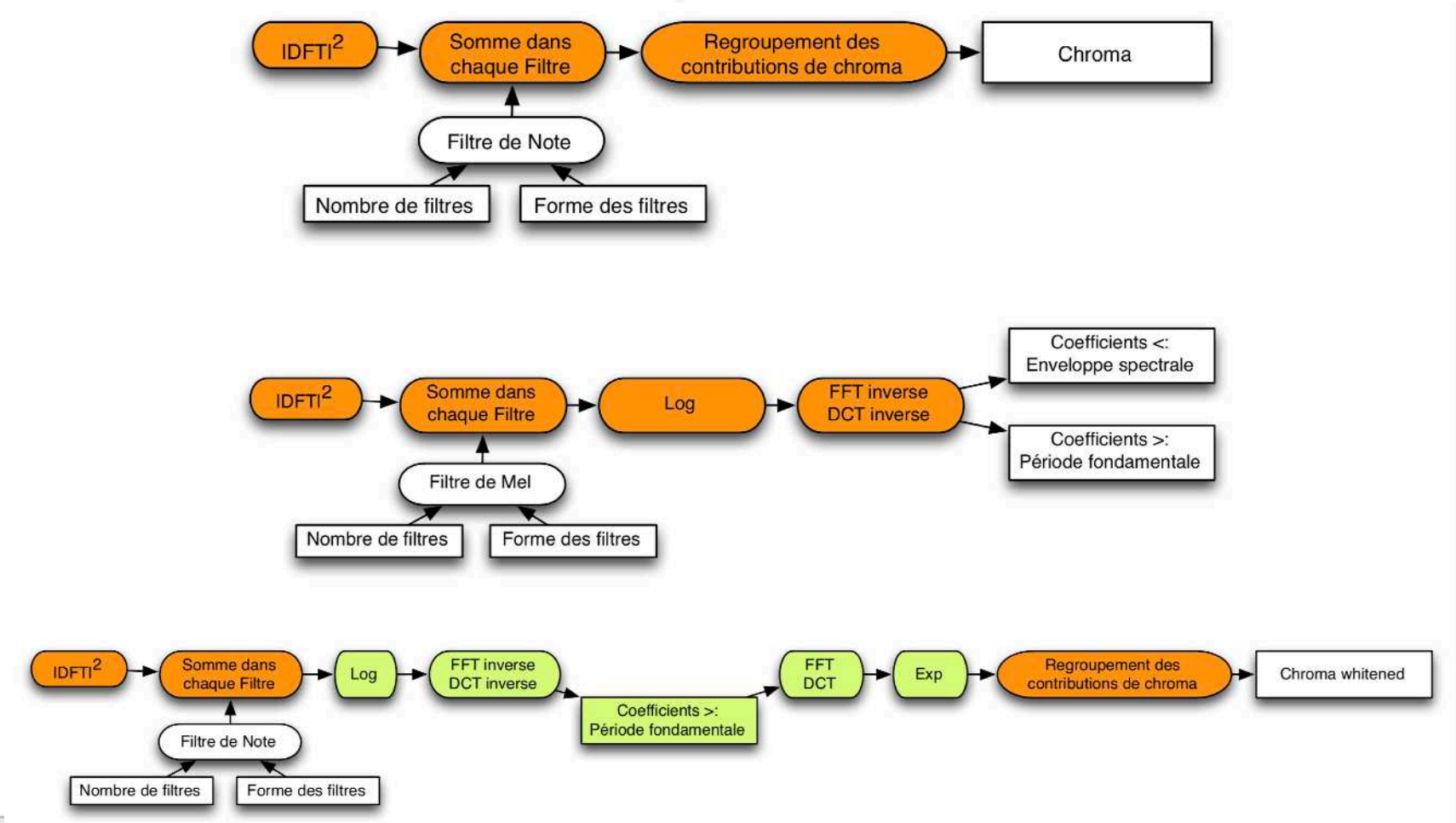


Exemple violon



Chroma/ Pitch Class Profile (PCP)

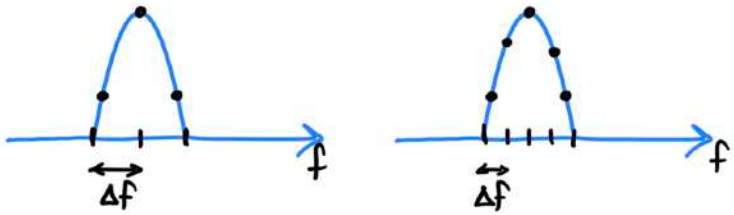
- **Variation of the computation: whitening**



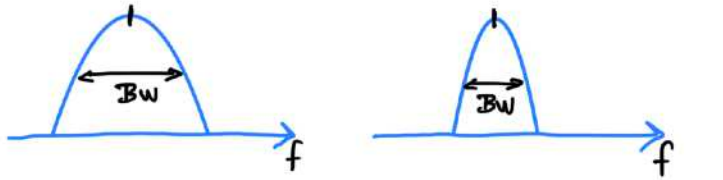
Constant-Q-Transform (CQT)

Constant-Q-Transform (CQT)

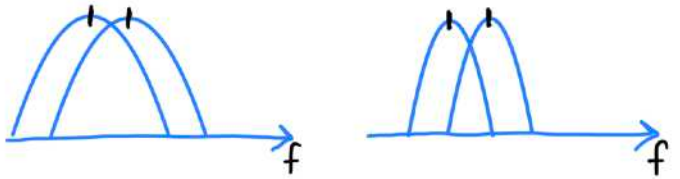
- Discrete Fourier Transform (DFT)
  - _ Definition : Spectral **precision** : $\Delta f = \frac{sr}{N}$
    - it is the step-size at which the Fourier spectrum is sampled
    - it depends on the size of the DFT: N
    - we can improve the precision by increasing N



  - _ Definition : Spectral **resolution** : $B_w = \frac{C_w}{L}$
    - it describes the ability to discriminate (separate in the spectrum) two adjacent simultaneous frequencies



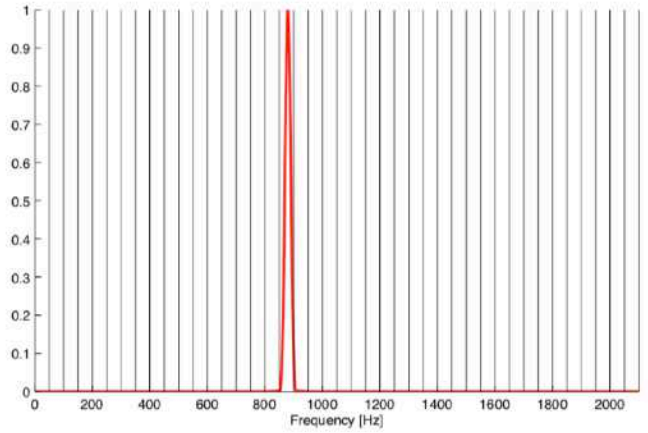
  - – Warning :
    - even if we increase N (zero-padding) while keeping L constant will not improve the resolution !



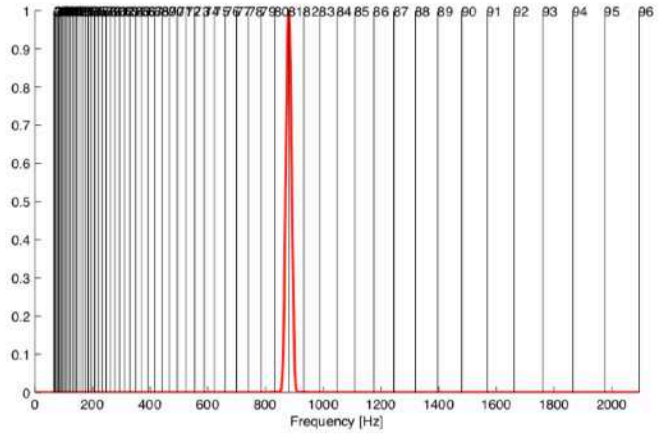
- In a DFT:
  - – Spectral precision and resolution are constant over frequencies

Constant-Q-Transform (CQT)

- In musical audio
  - the frequencies of the pitches are logarithmically spaced $f_k = f_0 \cdot 2^{\frac{k}{12}}$
    - if we choose A-4 = la-3 = 440 Hz as the reference
    - to go from midi-pitches m_k to frequencies f_k:
      - $f_k = 440 \cdot 2^{\frac{m_k - 69}{12}}$
    - to go from frequencies f_k to midi-pitches m_k:
      - $m_k = 12 \log_2 \frac{f_k}{440} + 69$
  - pitch frequencies are
    - close together in low frequencies,
    - distant in high frequencies
  - The **spectral resolution** of the DFT
    - is not sufficient (to separate adjacent notes) in low frequencies
    - is too large for high frequencies



Espacement linéaire de la DFT



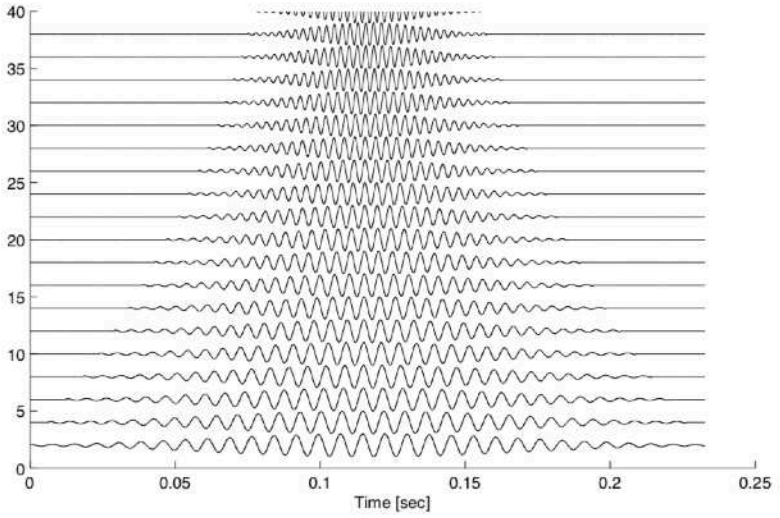
Espacement logarithmique des hauteurs de notes

Constant-Q-Transform (CQT)

- Solution ?
  - Change the spectral resolution B_w depending on the frequency f_k being considered

- How ?
  - By changing the window length L for eadh frequency f_k
  - The factor $Q = \frac{f_k}{f_{k+1} - f_k}$ should remains constant in frequency
    - $Q = \frac{f_k}{Bw} = \frac{f_k}{Cw/L} = \frac{f_k \cdot L}{Cw}$
  - We choose a different L for each frequency f_k
    - $L_k = \frac{Q \cdot Cw}{f_k}$

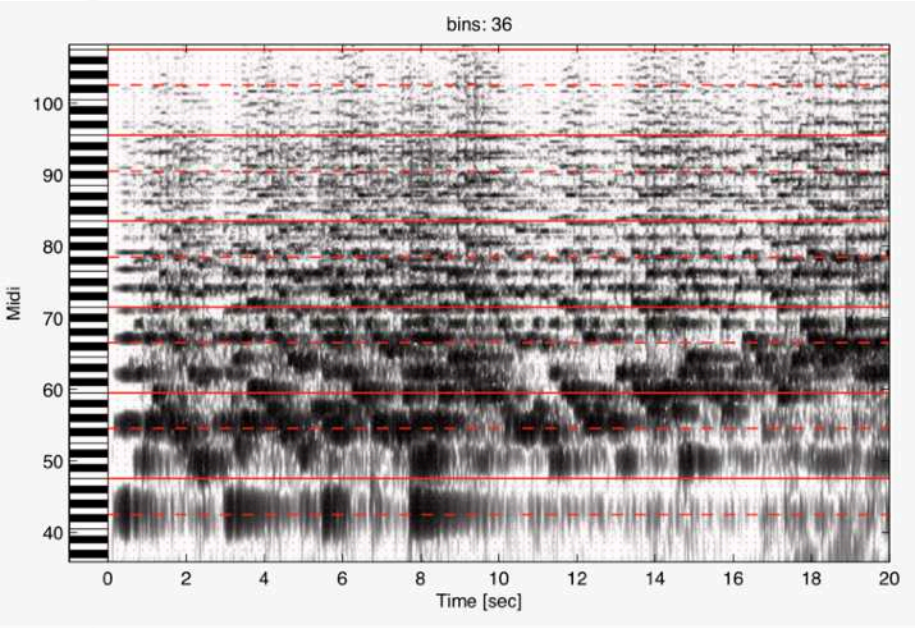

An efficient algorithm for the calculation of a constant Q transform



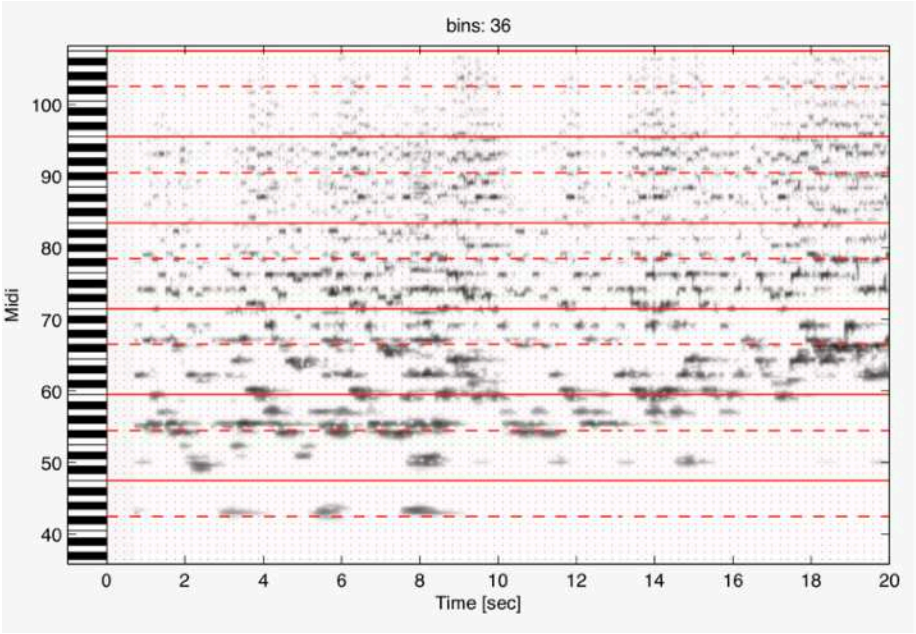
[J. Brown and M. Puckette. An efficient algorithm for the calculation of a constant q transform. JASA, 1992.]

Constant-Q-Transform (CQT)

Example (using DFT)

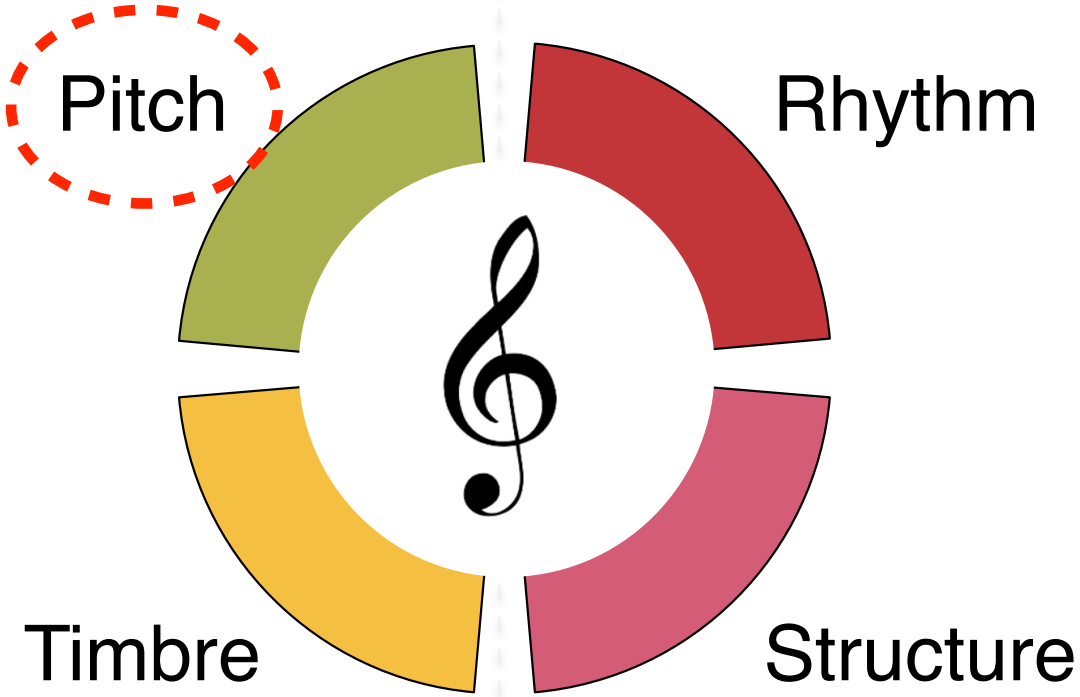


Example (using the CQT)



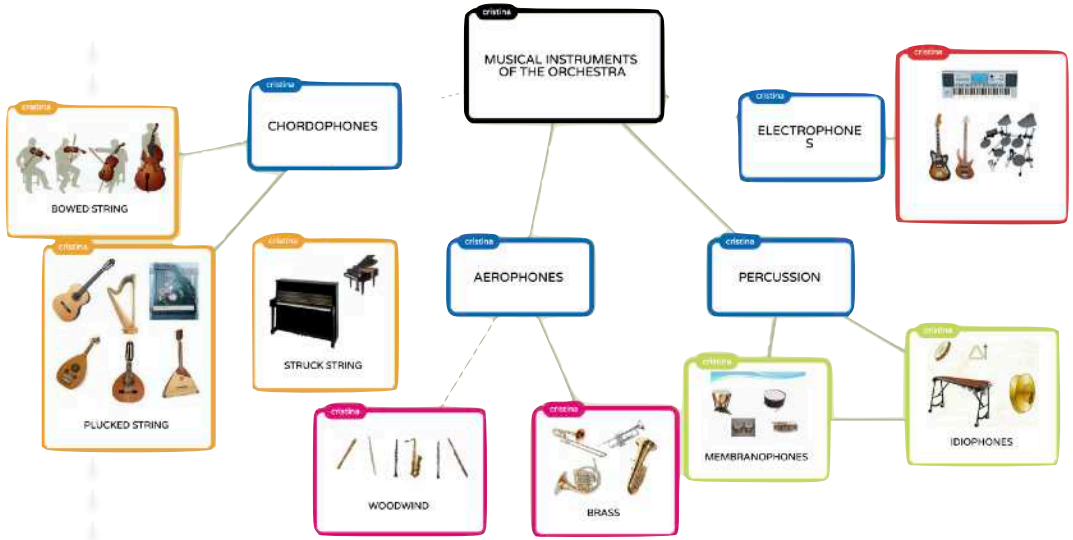
Reminder of Musical Concepts

What is pitch ?

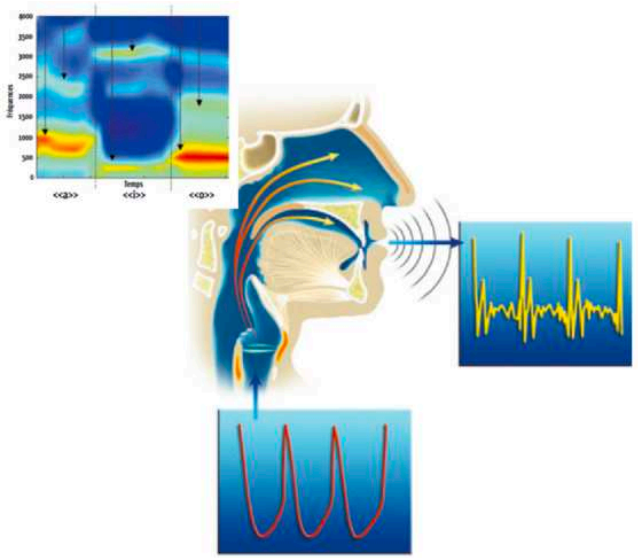


Harmonic sounds

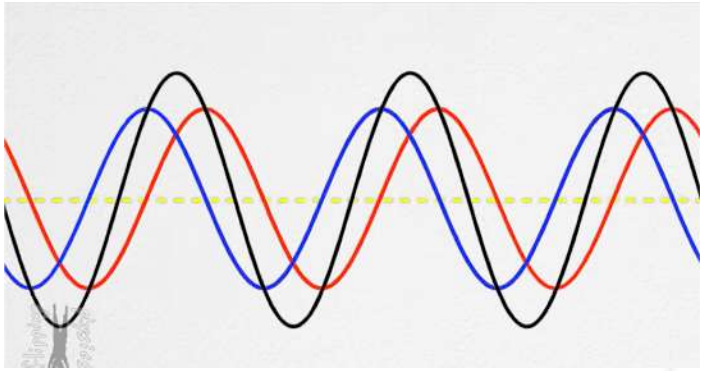
- Most musical instruments produce harmonic sounds
  - strings (plucked, bowed, hammered)
  - woodwinds, brass



- Why do they produce harmonic sounds ?
  - Two visions:
    - 1) periodic signal
    - 2) vibration modes



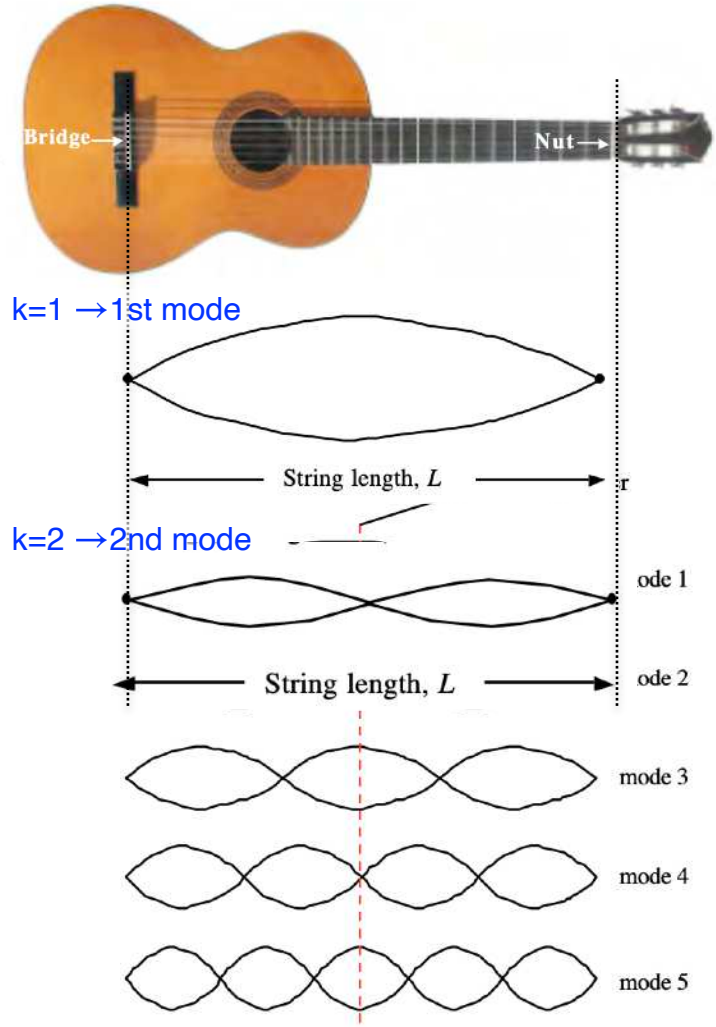
- **Reminder**
  - **Standing wave** in a string fixed at both ends



  - Possible **frequency modes** of a fingered string

$$f_k = k\frac{v}{\lambda} = k\frac{v}{2L} = k\frac{\sqrt{T/\mu}}{2L}$$

    - k vibration mode or harmonic number
    - T: string tension,
    - μ: linear mass density,



Harmonic sounds
Fourier series

- Harmonic representation of sound
  - Fourier series

$$f(t) = \sum_{n=0}^{+\infty} k_n \sin(n\omega_0 t + \phi_n)$$

$$\sin(a+b) = \sin(a)\cos(b) + \cos(a)\sin(b)$$

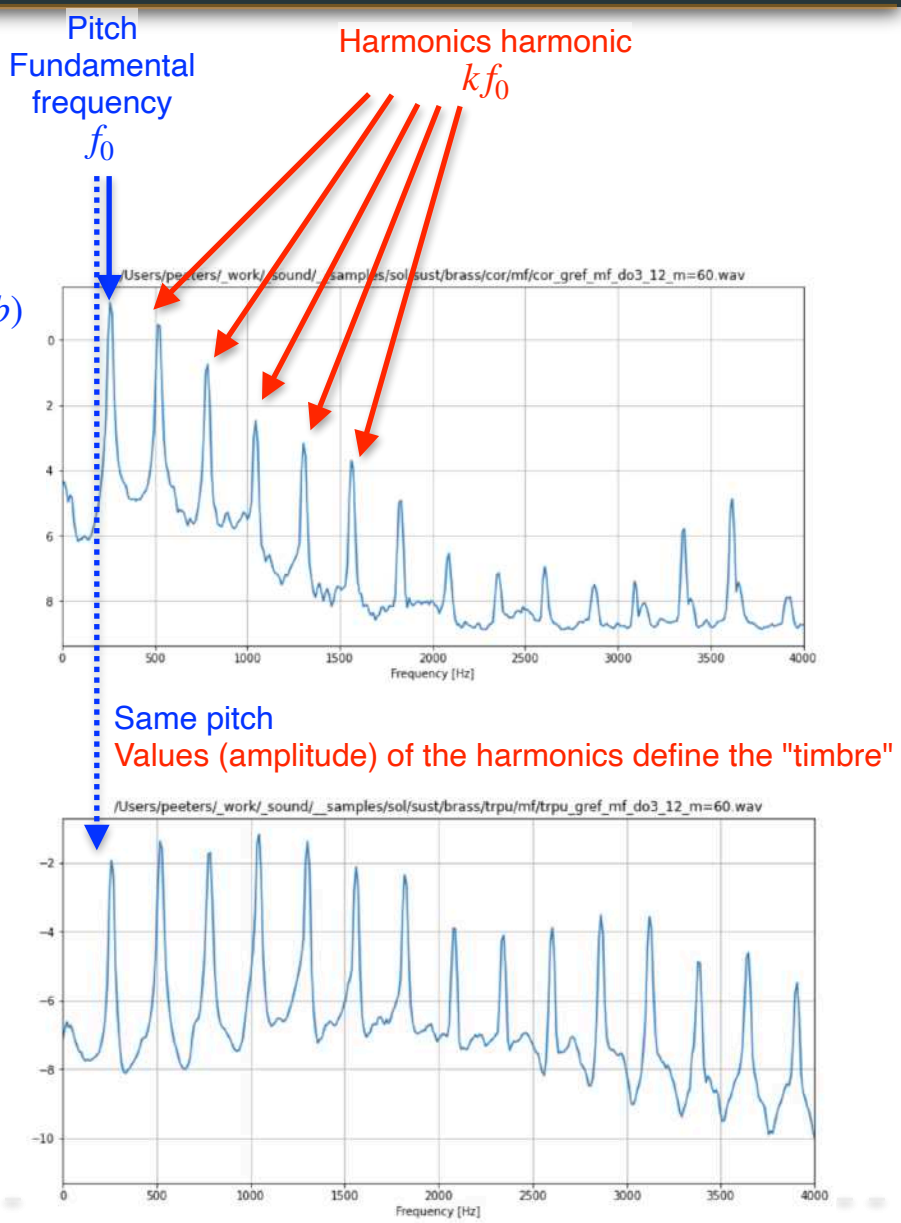
$$= \sum_{n=0}^{+\infty} A_n \cos(n\omega_0 t) + B_n \sin(n\omega_0 t)$$

$$= \sum_{n=-\infty}^{+\infty} c_n e^{jn\omega_0 t}$$

$$c_n = \begin{cases} \frac{A_n + iB_n}{2} & \text{if } n < 0 \\ \frac{A_0}{2} & \text{if } n = 0 \\ \frac{A_n - iB_n}{2} & \text{if } n > 0 \end{cases}$$

- Pitch (fundamental frequency)

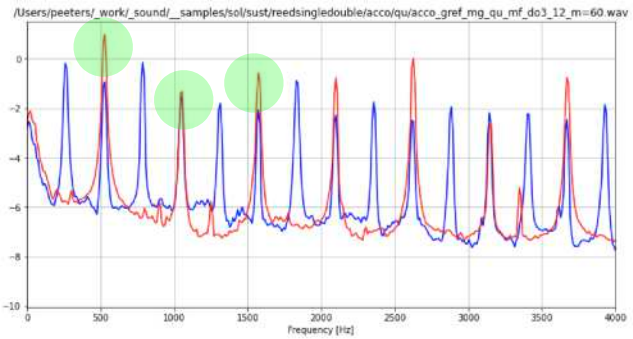
Mode of vibration	Frequency name (for any type of overtone)	Frequency name (for harmonic overtones)
First	Fundamental	First harmonic
Second	First overtone	Second harmonic
Third	Second overtone	Third harmonic
Fourth	Third overtone	Fourth harmonic



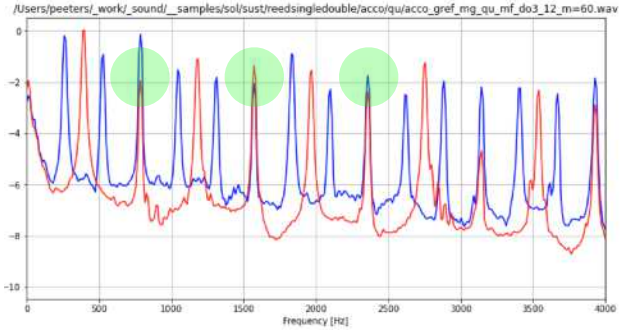
Harmonic sounds
Intervals

- Consonant/ Dissonant intervals between pitches
  - can be explained by the overlap between the harmonics
  - closeness between harmonics that will fall into the same critical band

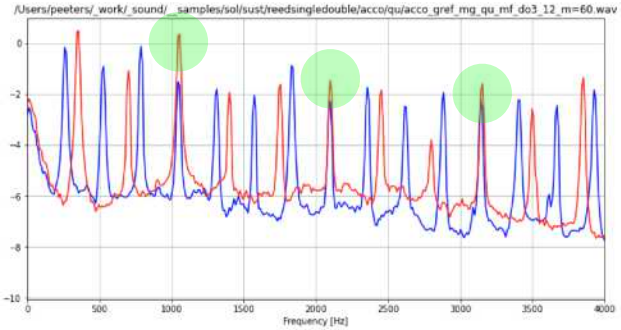
Octave interval ($F_0 = 2f_0$)



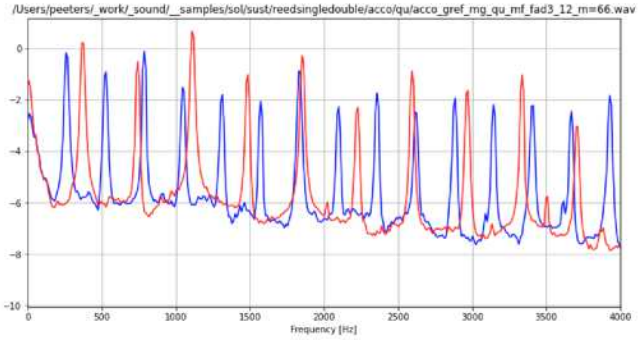
5th interval ($2F_0 = 3f_0$)



4th interval ($3F_0 = 4f_0$)



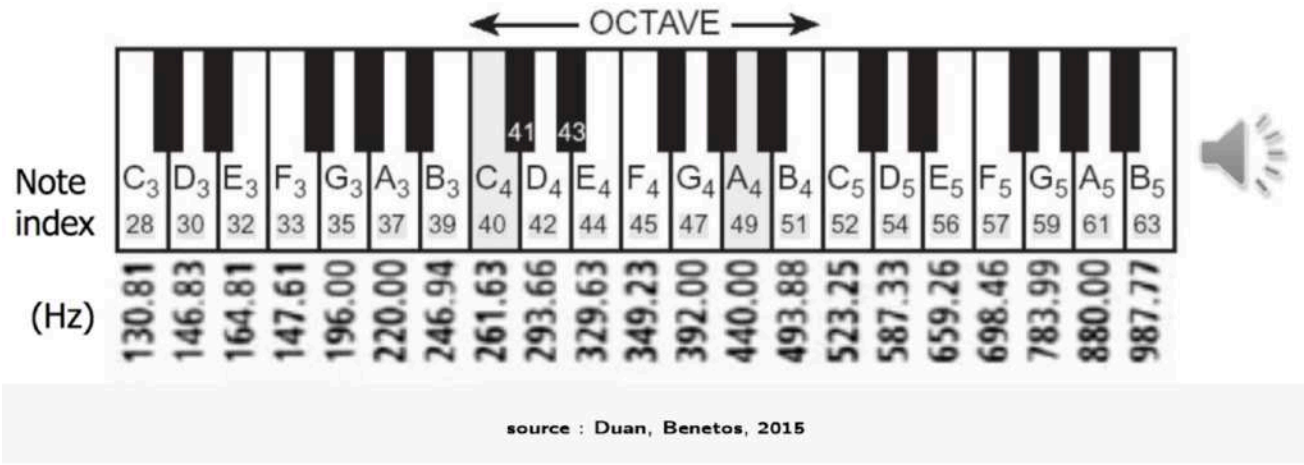
Triton (most dissonant !!!)



What is pitch ?

- **Pitch:**
  - the attribute of auditory sensation in terms of which sounds may be **ordered** on a scale extending from low to high (ANSI)
  - (Operational) A sound has a certain pitch if it can be reliably matched to a **sine tone** of a given frequency at 40 dB SPL
  - People hear pitch in a **logarithmic** scale

- **Notes =** Musical representation of pitch: do-3, re-3, mi-3

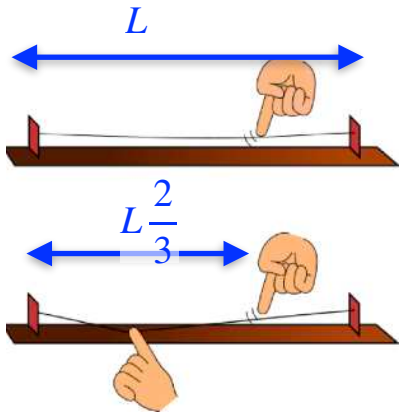


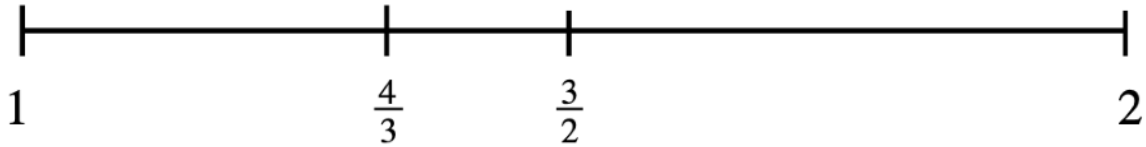
source : Duan, Benetos, 2015

What is pitch ?
Mapping Pitch to Notes

- To **map Pitch to Notes** we need to define an **organisation of the pitches**
  - **Temperament** = **tuning system** that allows representing (at best) the intervals of just intonation
  - Temperament refers to the various tuning systems for the subdivision of the octave
- **Intervals** ? the difference in pitch between two sounds
  - **Octave**: do-3 → do-4 (most consonant) ... because of the harmonic series ⇒ 2nd harmonics
  - **Fifth**: do-3 → sol-3 (very consonant) ... because of the harmonic series ⇒ 3rd harmonics
  - **Fourth**: do-3 → fa-3 (very consonant) ... because of the harmonic series ⇒ 4th harmonics
- The various **temperament** differs according to **how an octave is subdivided into notes** to allow representing the intervals

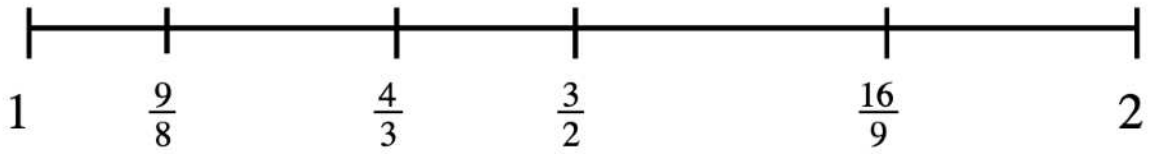
- **(1) Pythagorean temperament**
  - Pythagorean temperament = based on natural proportions **3:2**
  - Consider a string of length L, it produces a frequency $f = k\frac{v}{2L}$
  - Divide/Multiply the length by a factor 3:2
  - If we finger the string a $L' =$ each time we bring it back to main octave $\in [1,2]$
    - $L\left(\frac{1}{2}\right) \rightarrow 2f$ → octave (consonant interval) = **do**
    - $L\left(\frac{2}{3}\right) \rightarrow \frac{3}{2}f$ → 5th (consonant interval) = **sol**
    - $L\left(\frac{3}{2}\right) \rightarrow \frac{2}{3}f \rightarrow \frac{4}{3}f$ → 4th (consonant interval) = **fa**





- **(1) Pythagorean temperament**
  - If we finger the string a $L' =$ each time we bring it back to main octave $\in [1,2]$
    - $L\left(\frac{2}{3}\right) \rightarrow \frac{3}{2}f$ **=sol** $\quad L\left(\frac{2}{3}\right)^2 \rightarrow \frac{3^2}{2^2}f \rightarrow \frac{9}{8}f =$ **ré**
    - $L\left(\frac{3}{2}\right) \rightarrow \frac{2}{3}f \rightarrow \frac{4}{3}f$ **=fa** $\quad L\left(\frac{3}{2}\right)^2 \rightarrow \frac{2^2}{3^2}f \rightarrow \frac{16}{9}f =$ **si-b**
  -

Pentatonic scale !



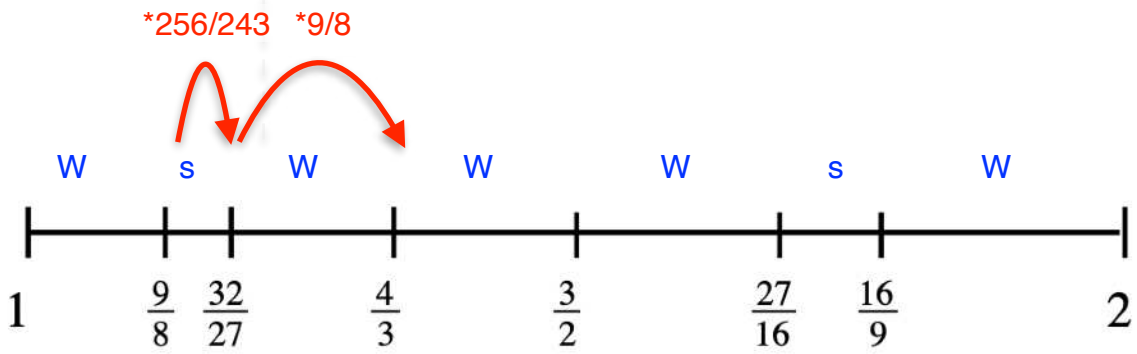
- **(1) Pythagorean temperament**
  - If we finger the string a $L' =$ each time we bring it back to main octave $\in [1,2]$
    - $L\left(\frac{2}{3}\right) \rightarrow \frac{3}{2} f$ **=sol**
    - $L\left(\frac{3}{2}\right) \rightarrow \frac{2}{3} f \rightarrow \frac{4}{3} f$ **=fa**

$L\left(\frac{2}{3}\right)^2 \rightarrow \frac{3^2}{2^2} f \rightarrow \frac{9}{8} f$ **=ré**

$L\left(\frac{3}{2}\right)^2 \rightarrow \frac{2^2}{3^2} f \rightarrow \frac{16}{9} f$ **=si-b**

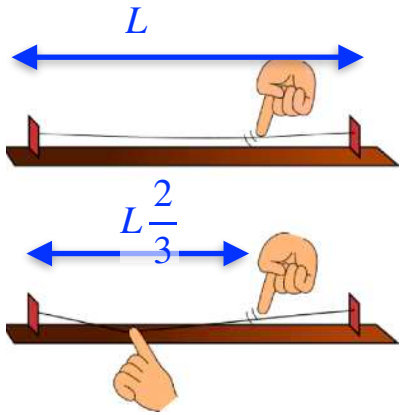
$L\left(\frac{2}{3}\right)^3 \rightarrow \frac{3^3}{2^3} f \rightarrow \frac{27}{16} f$ **=la**

$L\left(\frac{3}{2}\right)^3 \rightarrow \frac{2^3}{3^3} f \rightarrow \frac{32}{27} f$ **=mi-b**

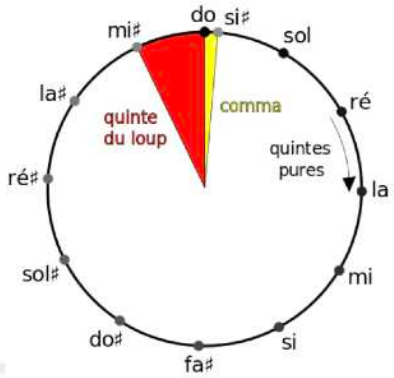


What is pitch ?
Temperaments

- **(1) Pythagorean temperament**
  - Pythagorean temperament = based on natural proportions 3:2
  - Consider a string of length L, it produces a frequency $f = k\frac{v}{2L}$
  - If we finger the string a $L' =$ each time we bring it back to main octave $\in [1,2]$
    - $L\left(\frac{1}{2}\right) \rightarrow 2f$ → octave (consonant interval) = **do**
    - $L\left(\frac{2}{3}\right) \rightarrow \frac{3}{2}f = 1.5f$ → 5th (consonant interval) = **sol**
    - $L\left(\frac{2}{3}\right)^2 \rightarrow \frac{3^2}{2^2} \rightarrow f\frac{3^2}{2^3}f \approx 1.13$ → 5th (5th) back to main octave (divide frequency by 2) = **ré**
    - $L\left(\frac{2}{3}\right)^3 \rightarrow \frac{3^3}{2^3}f \rightarrow \frac{3^3}{2^4}f \approx 1{,}69$ → 5th (5th (5th)) back to main octave = **la**
    - and so one



  - Resulting scale
    - problem: we are not getting back to do: $\frac{3^{12}}{2^{12}}\frac{1}{2^7} = 1.0136 \neq 1$
    - to get back to the starting note (do) we need to use a special fifth
  - Intervals are not equal !!!
    - not possible to transpose a given piece of music !!!
    - semi-tone intervals are not exactly half of the whole tones

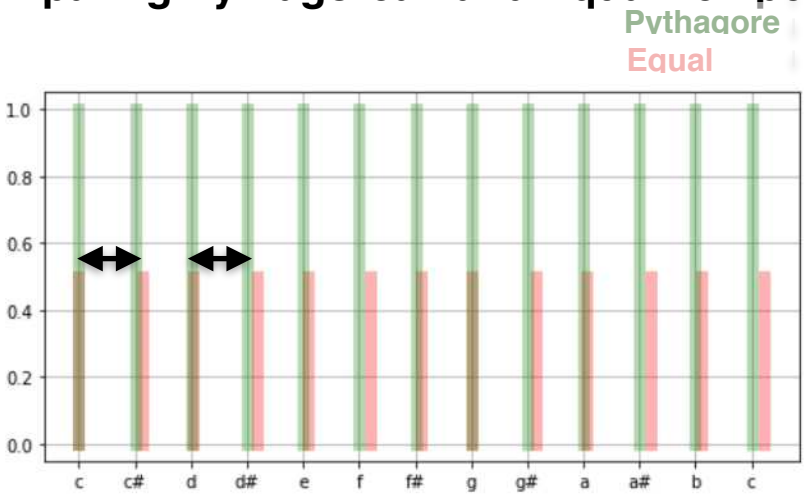


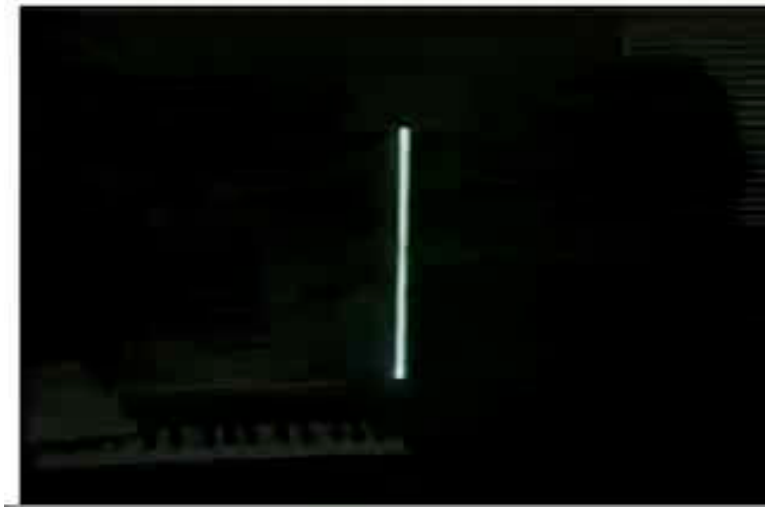
What is pitch ?
Temperaments

- **(2) Equal temperament**
  - divide an octave in 12 equal intervals →intervals: $\frac{f_1}{f_0} = 2^{\frac{1}{12}}$

$$f_0 = f \cdot 2^{\frac{0}{12}}$$
$$f_1 = f \cdot 2^{\frac{1}{12}}$$
$$f_2 = f \cdot 2^{\frac{2}{12}}$$
$$\cdots$$
$$f_{12} = f \cdot 2^{\frac{12}{12}} = 2 \cdot f_0$$

- **Comparing Pythagorean and Equal Temperament**





https://fr.wikipedia.org/wiki/Courbe_de_Lissajous

https://www.youtube.com/watch?v=6NII4No3s0M

What is pitch ?

- **Mapping Pitch to Notes** (using the equal temperament)

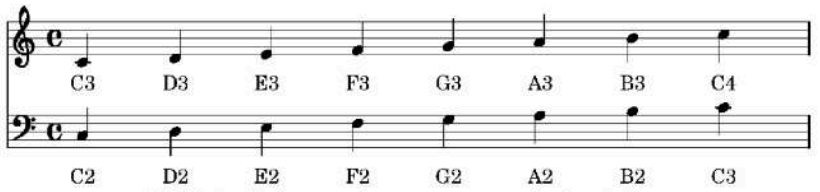
  - 261 Hz → C-4 or do-3 (do at 3rd octave)
  - 440 Hz → A-4 or la-3 (la at 3rd octave)
  - 880 Hz → A-5 or la-4 (la at 4th octave)

  – **midi values**

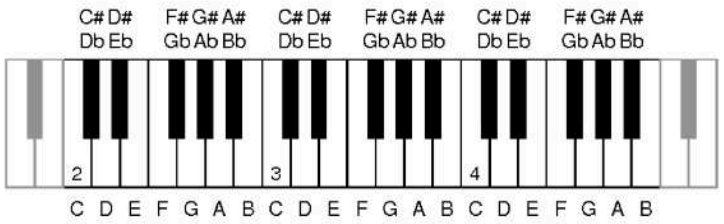
  - $f = 440 \cdot 2^{\frac{(m-69)}{12}}$
  - $m = 12 \log_2 \left(\frac{f}{440} \right) + 69$
  - 261 Hz → 60
  - 440 Hz → 69
  - 880 Hz → 81

Note	C3	D3	E3	F3	G3	A3	B3	C4
f_0, MIDI	48	50	52	53	55	57	59	60
f_0, Hz	130.8	146.8	164.8	174.6	196.0	220.0	246.9	261.6

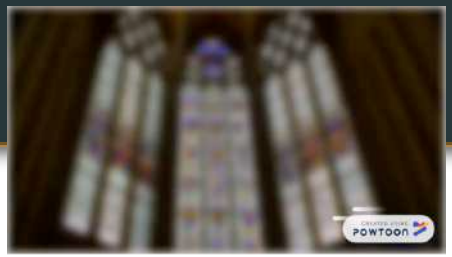
(a) Fundamental frequencies and MIDI note numbers for note names.



(b) Note pitches on the common musical notation.

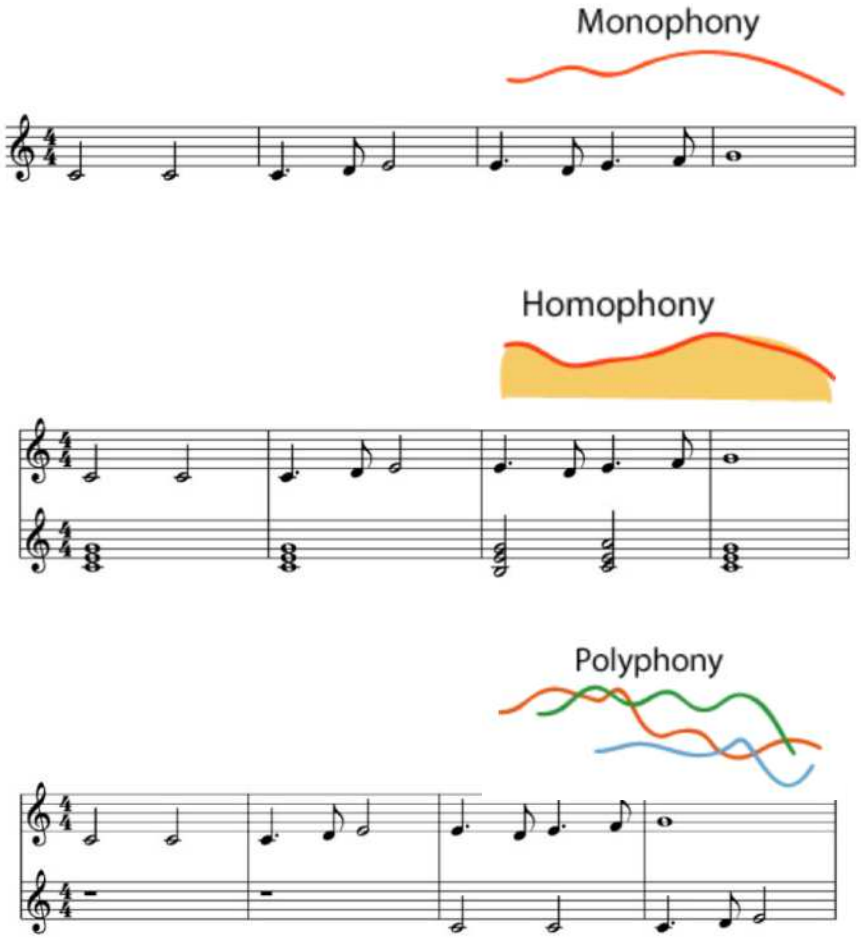


What is pitch ?
How pitches are organised ?



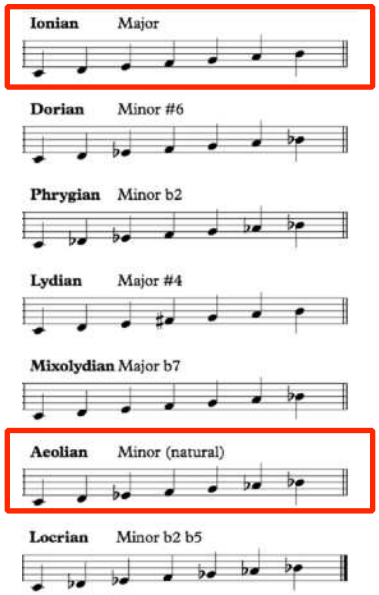
https://www.youtube.com/watch?v=e5Ip742VIyQ

- **Musical textures**
  - **Monophonic**
    - one voice, monophonic texture (monophony) refers to a single melodic line, though it may be played by one or many instruments
    - voices may be in exact unison or in different octaves, as long as the same notes and rhythms are played
  - **Homophonic**
    - one voice, a melody, which stands out from background accompaniment
    - accompaniment may be simple chords or a harmony with melodic interest, but in either case, the main melody must be clearly distinguishable
  - **Polyphonic**
    - or counterpoint involves multiple melodic voices, all of equal importance, occurring simultaneously.
    - complex, dense texture is typical of Renaissance and baroque music

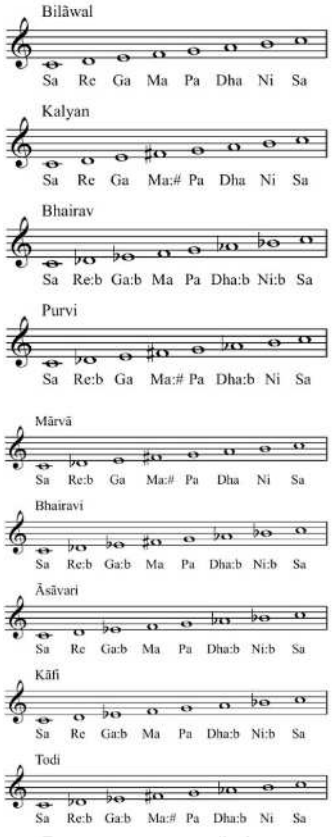


What is pitch ?
How pitches are organised ?

- **Modal music**
  - since the Greeks, Gregorian and Church
  - which uses modal scales
    - (as opposed to Major/minor scales)
  - centred on horizontal-melody often monophonic
    - (as opposed to vertical-harmony),
  - in other cultures:
    - Indian music ("râgas"), Turkish music ("maqâm"), or modal jazz (Miles Davis "Kind of Blue")



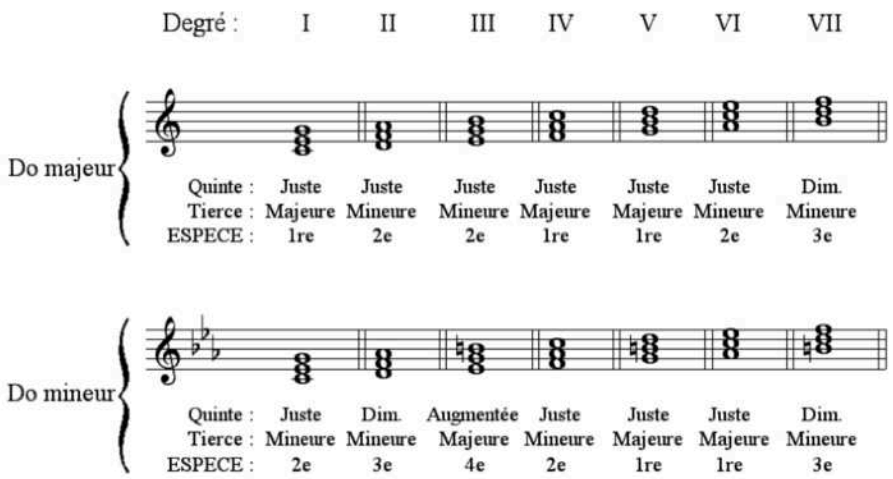
Greek, Gregorian, Church modes

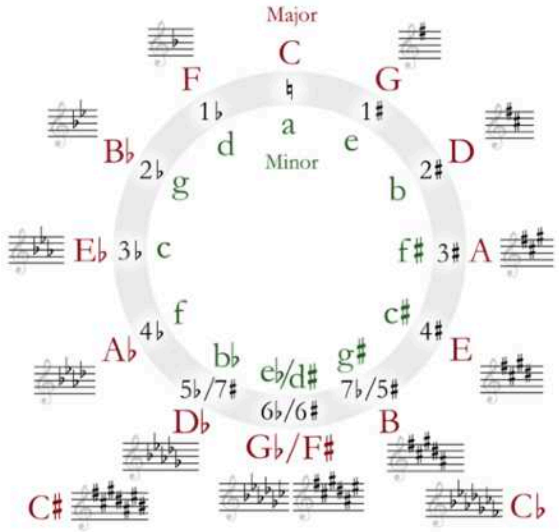


Indian raga

What is pitch ?
How pitches are organised ?

- **Tonal music**
  - centred around the tonality
  - define the set of relationships between notes, structured chords around a given tonality
  - tonal language is built upon the diatonic Major and minor scales by applying the ton harmony
  - Polyphonic techniques
    - horizontal: poly-melody: counter-point
    - vertical: chords, succession of harmony
  - Major and minor scales
  - Circle of fifth
    - 12 Major and minor keys

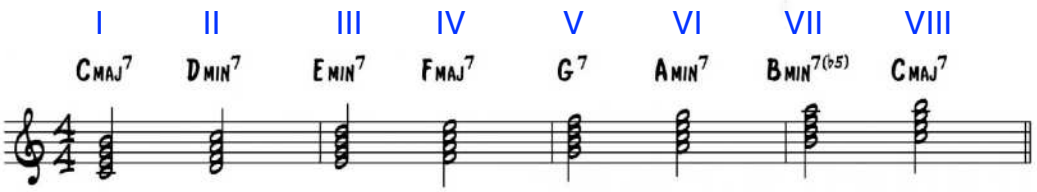




What is pitch ?
How pitches are organized ?

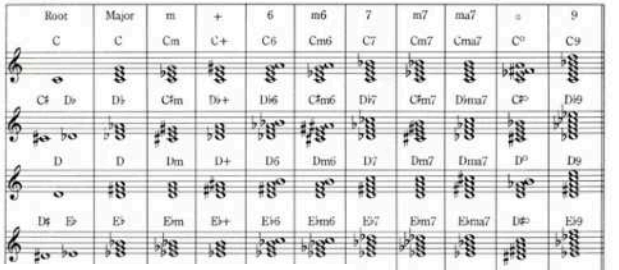
- **Chords:**
  - set of notes considered as a whole in terms of harmony
  - most of often, notes are played (almost) simultaneously
    - but also can be arpeggiated

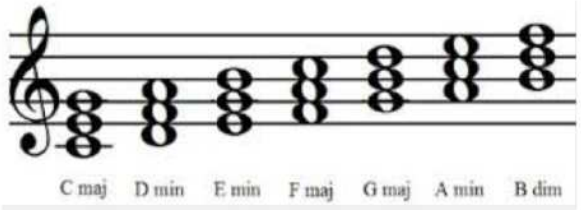
- Chords on Major scale



- Most common chord sequences
  - V → I
  - II → V → I
  - Anatole
  - The magic 4-chords succession in pop-music
    - I → V → VI → IV
    - VI → IV → I → V

Chord dictionary





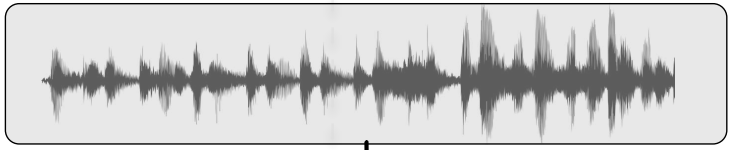
Root-note	Type of the chord
c, c#, d, d# … b	**Triads**: major (C-M: c,e,g) , minor (C-m: c, e♭ , g), suspended (C-sus2: c, d, g / C-sus4: c, f, g), augmented (C-aug: c, e, g#), diminished (C-dim: c, e♭, g♭)
	Tetrads: major 7 (C-M7: c, e, g, b), minor 7 (C-m7: c, e♭, g, b♭), dominant 7 (C-7: c, e, g, b♭), major 6 (C-M6: c, e, g, a), minor 6 (C-m6: c, e♭, g, a) …
	Pentads: major 9 (C-M9: c, e g, b, d), dominant 9 (C-9: c, e, g, b♭, d) …

Automatic Chord Recognition (ACR)

Automatic Chord Recognition (ACR)

audio / music

input



Automatic Chord Recognition (ACR) System

sequence of chords

A A AAA AF#
I need another story

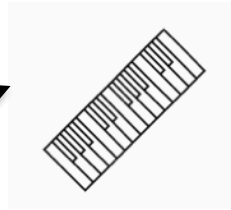
AA A A A E E
Something to get off my chest

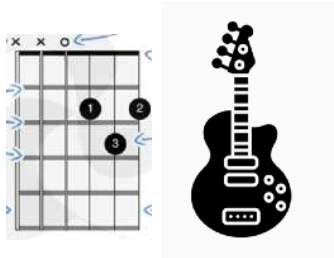
A A A AA AF#
My life gets kinda boring

A AA A A D DB
Need something that I can confess

A A B D E C# D
'Til all my sleeves are stained red

A A B C# D C# C#
From all the truth that I've said





Chord dictionary, chord vocabulary

- **En-harmonicity**:
  - we suppose that C# is the same as Db
- Different chord **types**:
  - triads (3 notes)
    - Major, minor, diminished, augmented
  - tetrads (4 notes)
    - Seventh, Sixth
  - extended chords
    - ...
- **Missing** notes
- Chord **inversions**
  - which one is the bass note ?
    - `(1,3,5)/1` → `(1,3,5)/3`

Chord Type		Shorthand Notation	Components List
Triad Chords:			
	Major	`maj`	`(3,5)`
	Minor	`min`	`(b3,5)`
	Diminished	`dim`	`(b3,b5)`
	Augmented	`aug`	`(3,#5)`
Seventh Chords:			
	Major Seventh	`maj7`	`(3,5,7)`
	Minor Seventh	`min7`	`(b3,5,b7)`
	Seventh	`7`	`(3,5,b7)`
	Diminished Seventh	`dim7`	`(b3,b5,bb7)`
	Half Diminished Seventh	`hdim7`	`(b3,b5,b7)`
	Minor (Major Seventh)	`minmaj7`	`(b3,5,7)`
Sixth Chords:			
	Major Sixth	`maj6`	`(3,5,6)`
	Minor Sixth	`min6`	`(b3,5,6)`
Extended Chords:			
	Ninth	`9`	`(3,5,b7,9)`
	Major Ninth	`maj9`	`(3,5,7,9)`
	Minor Ninth	`min9`	`(b3,5,b7,9)`
Suspended Chords:			
	Suspended 4th	`sus4`	`(4,5)`

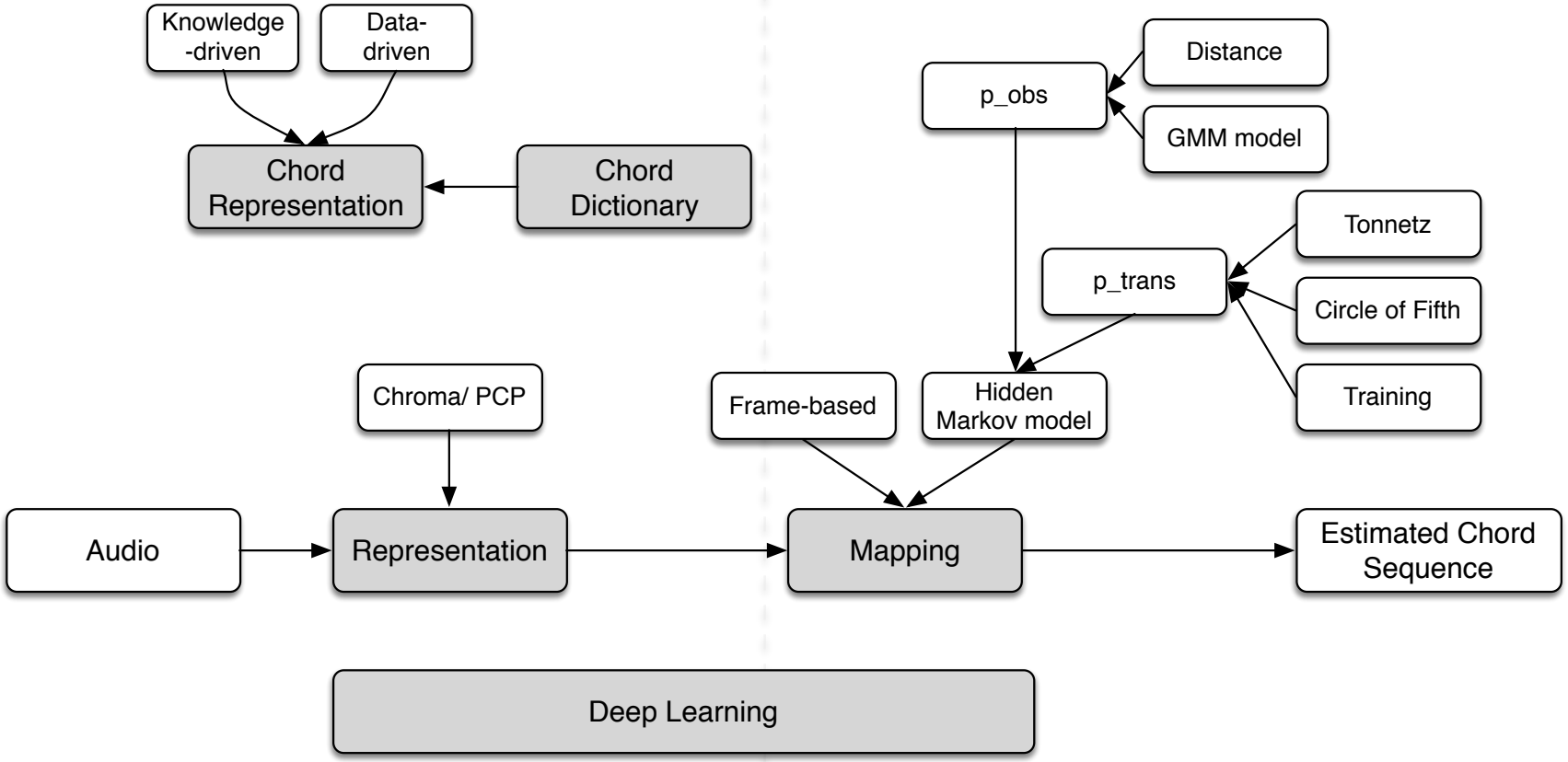
```
C:min7    ≡    C:(b3,5,b7)
C:min7(*5,11)    ≡    C:(b3,b7,11)
C    ≡    C:maj    ≡    C:(3,5)
A/3    ≡    A:maj/3    ≡    A:(3,5)/3
C:maj(4)    ≡    C:(3,4,5)
```

Harte, C., M. Sandler, S. Abdallah, and E. Gómez. 2005. "Symbolic representation of musical chords: A proposed syntax for text annotations." In *Proceedings of the 6th International Society for Music Information Retrieval Conference (ISMIR)*, 66–71.

Automatic Chord Recognition (ACR) Systems

Automatic Chord Recognition (ACR) Systems

Automatic Chord Recognition System



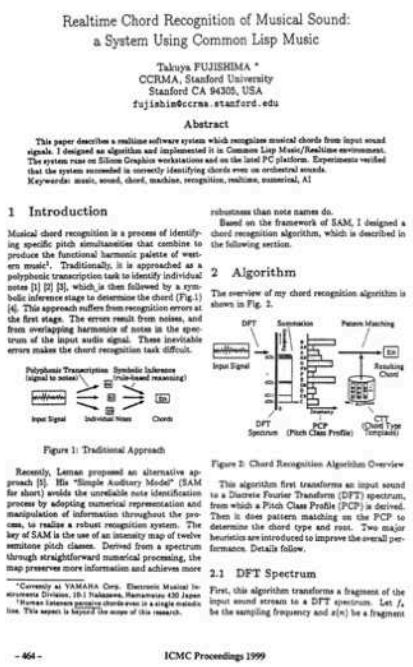
Automatic Chord Recognition (ACR) Systems

Brief overview of system evolution

- *as usual, the **first systems** define the task, the performance measures, and provide a first test-set; **later systems** deals with scalability issues and create large test-set; **current systems** use this large dataset to train systems using deep-learning*
- Frame-based/ template-based approach
  - **1999** → T. Fujishima. "Realtime chord recognition of musical sound: a system using common lisp music". In Proc. of ICMC,1999.
- Hidden-Markov-Model (HMM) based approaches
  - **2003** → A. Sheh and D. P. W. Ellis. "Chord segmentation and recognition using em-trained hidden Markov models". In Proc. of ISMIR, 2003
  - **2007** → H. Papadopoulos and G. Peeters. "Large-scale study of chord estimation algorithms based on chroma representation". In Proc. of IEEE CBMI, 2007
- Splitting into bass/middle/chroma
  - **2012** → Yizhao Ni, Matt McVicar, Raul Santos-Rodriguez, and Tijl De Bie. "An end-to-end machine learning system for harmonic analysis of music". IEEE TASLP, 2012.
- Deep learning approaches
  - **2013** → Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. "Audio chord recognition with recurrent neural networks". In ISMIR, 2013
  - **2016** → Filip Korzeniowski and Gerhard Widmer. "Feature learning for chord recognition: the deep chroma extractor". In ISMIR, 2016.
  - **2017** → B. McFee and J. P. Bello. "Structured training for large-vocabulary chord recognition". In Proc. of ISMIR, 2017
  - **2021** →C. Weiß and G. Peeters. "Training deep pitch-class representations with a multi-label CTC loss". In Proc. of ISMIR, 2021

Automatic Chord Recognition (ACR) Systems

(1) Frame-based/ template-based approach



Realtime Chord Recognition of Musical Sound:
a System Using Common Lisp Music

Takuya FUJISHIMA *
CCRMA, Stanford University
Stanford CA 94305, USA
fujishim@ccrma.stanford.edu

Abstract

This paper describes a realtime software system which recognizes musical chords from input sound signals. I designed an algorithm and implemented it in Common Lisp Music/Realtime environment. The system runs on Silicon Graphics workstations and on the latest PC platform. Experiments verified that the system succeeded in correctly identifying chords even on orchestral sounds.
Keywords: music, sound, chord, machine, recognition, realtime, numerical, AI

1 Introduction

Musical chord recognition is a process of identifying specific pitch simultaneities that combine to produce the functional harmonic palette of western music[1]. Traditionally, it is approached as a polyphonic transcription task to identify individual notes [1] [2] [3], which is then followed by a symbolic inference stage to determine the chord (Fig.1) [4]. This approach suffers from recognition errors at the first stage. The errors result from noises, and from overlapping harmonics of notes in the spectrum of the input audio signal. These inevitable errors makes the chord recognition task difficult.

Figure 1: Traditional Approach

Recently, Leman proposed an alternative approach [5]. His "Simple Auditory Model" (SAM for short) avoids the unreliable note identification process by adopting numerical representation and manipulation of information throughout the process, to realize a robust recognition system. The key of SAM is the use of an intensity map of twelve semitone pitch classes. Derived from a spectrum through straightforward numerical processing, the map preserves more information and achieves more robustness than note names do.

Based on the framework of SAM, I designed a chord recognition algorithm, which is described in the following section.

2 Algorithm

The overview of my chord recognition algorithm is shown in Fig. 2.

Figure 2: Chord Recognition Algorithm Overview

This algorithm first transforms an input sound to a Discrete Fourier Transform (DFT) spectrum, from which a Pitch Class Profile (PCP) is derived. Then it does pattern matching on the PCP to determine the chord type and root. Two major heuristics are introduced to improve the overall performance. Details follow.

2.1 DFT Spectrum

First, this algorithm transforms a fragment of the input sound stream to a DFT spectrum. Let f_s be the sampling frequency and $x(n)$ be a fragment

*Currently at YAMAHA Corp. Electronic Musical Instruments Division, 10-1 Nakazawa, Hamamatsu 430 Japan
†Human listeners perceive chords even in a single melodic line. This aspect is beyond the scope of this research.

of the input sound signal with N samples, where $n = 0, 1, \cdots N-1$ is the index. Then the DFT spectrum $X(k)$ is given for $k = 0, 1, ..., N-1$ as:

$$X(k) = \sum_{n=0}^{N-1} e^{-2\pi i k n / N} \cdot x(n)$$

Note that $X(0), \cdots, X(N/2-1)$, or the first half, gives the entire spectrum, as $x(n)$ is real.

2.2 Pitch Class Profile

From $X(k)$, this algorithm generates a Pitch Class Profile (PCP). A PCP is similar to the intensity map in Leman's SAM. It is a twelve dimension vector which represents the intensities of the twelve semitone pitch classes. For instance, the first PCP element shows how strong C is in total.

Let $PCP(p)$ be a PCP. It is defined for the index $p = 0, 1, \cdots, 11$ as:

$$PCP(p) = \sum_{i \text{ s.t. } M(i)=p} \|X(i)\|^2$$

where $M(i)$ is a table which maps a spectrum bin index to the PCP index. $M(i)$ can initially be defined as:

$$M(i) = \begin{cases} -1 & \text{for } i = 0 \\ \text{round}(12 \log_2(i f_s \cdot \frac{1}{N}) / f_{ref}) \bmod 12 & \text{for } i = 1, 2, \cdots, N/2-1 \end{cases}$$

where f_{ref} is the reference frequency that falls into $PCP(0)$. The term $(f_s \cdot \frac{1}{N})$ represents the frequency of the spectrum bin $X(i)$. Note that $PCP(p)$ and $M(i)$ are actually further modified here for later use (see 2.4).

2.3 Pattern Matching

For each PCP obtained, this algorithm does pattern matching using the Chord Type Templates (CTT). The CTT's represent the knowledge of musical chords. $CTT_c(p)$ is set to 1 if the chord type c at root C has the pitch class p in it, and 0 otherwise. For instance, the CTT for "M" is $(1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0)$.

I chose 27 chords (Table 1) in 2 groups, based on if a chord stays within an octave([S1]) or not([S2]).

Two matching methods are compared: the nearest neighbor and the weighted sum[6]. Each calculates scores for all the chord types, and for all the roots rotating the PCP, to choose the one with the best score.

Nearest Neighbor Method

The score to be minimized for a chord type c is given as:

$$Score_{nearest,c} = \sum_{p=0}^{11} (T_c(p) - PCP(p))^2$$

Table 1: Chord Types Chosen

group	chord types
S1	C_{major}, C_{minor}, C_{dim7}, C_{dim7}, C_{dim7}, C_{dim7}, C_{m}, C_{M7}, C_{m7}, C_{7}, C_{7}, C_{M7}, C_{aug}, C_{m7}, C_{mM7}, C_{6}, C_{m6}, C_{sus4}
S2	C_{m9}, C_{7}, C_{9}, C_{7-5}, $C_{m/G}$, $C_{M7/G}$, $C_{/G}$, $C_{7/G}$, $C_{9/G}$, $C_{7-5/G}$, $C_{7sus4/G}$, C_{sus}

It is put as a placeholder for readability.

where $T_c(p)$ is a typical PCP tuned by hand, starting from the original $CTT_c(p)$.

Weighted Sum Method

The score to be maximized is given as:

$$Score_{weighted,c} = \sum_{p=0}^{11} W_c(p) \cdot PCP(p)$$

where $W_c(p)$ is the weight vector tuned by hand, again starting from $CTT_c(p)$. Through the tuning, different chord types get different weights, so as to reflect the number of the notes in the chord type and the probability of the emergence of the chord type. Also, negative weights are put for better separation among similar chord types.

2.4 Heuristics

To improve the overall performance of this algorithm, I introduce the following two heuristics.

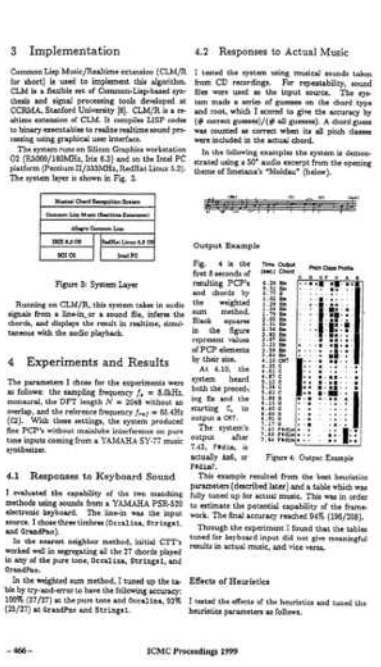
1) PCP Smoothing over time: Assuming that a chord usually lasts for several PCP frames, this algorithm does smoothing over past PCP's by an averaging operation to reduce the noise.

2) Chord Change Sensing: A chord change is usually sudden, when the PCP vector changes its direction in the twelve dimension space. Therefore, we can sense chord changes by monitoring the direction of the PCP vector.

Other minor heuristics include:

- PCP preprocessing, including non-linear scaling
- Elimination of less important regions[7] in $M(i)$
- Use of an DFT window
- Silence detection to avoid unnecessary analysis
- Attack detection to avoid noisy PCP's

for which details are not mentioned further here.



3 Implementation

Common Lisp Music/Realtime extension (CLM/R for short) is used to implement this algorithm. CLM is a flexible set of Common-Lisp-based synthesis and signal processing tools developed at CCRMA, Stanford University [8]. CLM/R is a realtime extension of CLM. It compiles 2.1ISP codes to binary executables to realize realtime sound processing using graphical user interface.

The system runs on Silicon Graphics workstation O2 (R5000/180MHz, Irix 6.3) and on the Intel PC platform (Pentium II/333MHz, RedHat Linux 5.2). The system layer is shown in Fig. 3.

Figure 3: System Layer

Running on CLM/R, this system takes in audio signals from a line-in or a sound file, infers the chords, and displays the result in realtime, simultaneous with the audio playback.

4 Experiments and Results

The parameters I chose for the experiments were as follows: the sampling frequency $f_s = 8.0$kHz, monaural, the DFT length $N = 2048$ without an overlap, and the reference frequency $f_{ref} = 61.4$Hz (C2). With these settings, the system produced five PCP's without unstable interference on pure tone inputs coming from a YAMAHA SY-77 music synthesizer.

4.1 Responses to Keyboard Sound

I evaluated the capability of the two matching methods using sounds from a YAMAHA PSR-520 electronic keyboard. The line-in was the input source. I chose three timbres (Vocalina, Stringsl, and GrandPno).

In the nearest neighbor method, initial CTT's worked well in segregating all the 27 chords played in any of the pure tone, Vocalina, Stringsl, and GrandPno.

In the weighted sum method, I tuned up the table by try-and-error to have the following accuracy: 100% (27/27) at the pure tone and Vocalina, 93% (25/27) at GrandPno and Stringsl.

4.2 Responses to Actual Music

I tested the system using musical sounds taken from CD recordings. For repeatability, sound files were used as the input source. The system made a series of guesses on the chord type and root, which I scored to give the accuracy by (# correct guesses)/(# all guesses). A chord guess was counted as correct when its all pitch classes were included in the actual chord.

In the following examples the system is demonstrated using a 50" audio excerpt from the opening theme of Smetana's "Moldau" (below).

Output Example

Figure 4: Output Example

Fig. 4 is the first 8 seconds of resulting PCP's and chords by the weighted sum method. Black squares in the figure represent values of PCP elements by their size.

At 4.10, the system heard both the preceding fis and the starting C_7 to output a cM7.

The system's output after 7.42, F#dim, is actually Am, or F#dim7.

This example resulted from the best heuristics parameters [described later] and a table which was fully tuned up for actual music. This was in order to estimate the potential capability of the framework. The final accuracy reached 94% (136/208).

Through the experiment I found that the tables tuned for keyboard input did not give meaningful results in actual music, and vice versa.

Effects of Heuristics

I tested the effects of the heuristics and tuned the heuristics parameters as follows.

Fig. 5 shows how the heuristics affects the PCP's for three cases. In each graph, the time goes from left to right.

a) Without the heuristics, PCP's are quite noisy.

b) "Smoothing over Time" reduces noise, but it oversmoothes, blurring chord change points.

c) "Chord Change Sensing" helps preserve the chord change points.

Figure 5: Effects of Heuristics

Fig. 6 is an accuracy plot for various heuristics parameters.

Figure 6: Accuracy v.s. Heuristics Parameters

Fig. 6 shows that the best result occurs at smoothing level = 0.5, chord change sensing = 2.0 which were actually used in the example in Fig. 4.

5 Discussion

The experiments show that this realtime system can successfully recognize musical chords at the signal level. It assigns correct chord names even in complex multi-timbral situations like orchestral music. Given the size of the system (1371 lines in LISP source code), this is a remarkable performance.

The PCP proves to be a reasonable internal representation. It conveys information sufficient to determine the chord type and root, though it discards the bass note information. The chords which cause "aliasing" in a PCP (52 in Table 1) can be recognized correctly.

Both the weighted sum method and the nearest neighbor method can correctly segregate 93% through 100% of the chords played with the pure tones or with synthesized sounds. On actual music, the weighted sum method can recognize chords with a high accuracy of 94%, while the nearest neighbor method results in insufficient accuracy. Therefore, these classifiers are sufficient at least in the ideal input situation, but better classifiers may be necessary to process actual music input.

The two heuristics for edge-preserving smoothing improve the overall accuracy by about 10%.

Future work will include a better classifier, the table tuning technique, bass note tracking, use of musical context for the symbol level error correction.

6 Acknowledgement

This research was done under direction of Professor Jonathan Berger at CCRMA, Stanford University. I appreciate all of the supports and fruitful discussions. I would also express my gratitude to Mr. Masatada Wachi, the director of board in YAMAHA Corporation, who gave me the opportunity of doing this research.
U.S. Patent Pending: Serial No. 09-281126

References

[1] Chris Chafe, David Jaffe, Kyle Kashima, Bernard Mont-Reynaud, Julius Smith, *Techniques for Note Identification in Polyphonic Music*, STAN-M-29, CCRMA, Stanford University Oct. 1985.

[2] Chris Chafe and David Jaffe, *SOURCE SEPARATION AND NOTE IDENTIFICATION IN POLYPHONIC MUSIC*, STAN-M-34, CCRMA, Stanford University Apr. 1986.

[3] Kunio Kashino, Kazuhiro Nakadai, Tomoyoshi Kinoshita and Hidekiko Tanaka, *Application of Bayesian Probability Network to Music Scene Analysis*, Proceedings of IJCAI 1995, 1995.

[4] Yoshi Asano, Haruhiro Katayose, and Seiji Inokuchi, *A Real-time Session Composer with Acoustic Polyphonic Instruments*, Proceedings of ICMC 1998, pp. 236-238, 1998.

[5] Marc. Leman, *Music and Schema Theory*, Springer-Verlag

[6] Donald R. Tveter, *The Pattern Recognition BASIS OF ARTIFICIAL INTELLIGENCE*, IEEE COMPUTER SOCIETY.

[7] Michele Biasutti, *Sharp low- and high-frequency limits on musical chord recognition*, Hearing Research, No.103, pp. 77-84, 1997.

[8] Bill Schottstaedt, *Machine Tongues XVII: CLM: Music V Meets Common Lisp*, Computer Music Journal, Vol.18, No.2, pp.30-38, 1994.

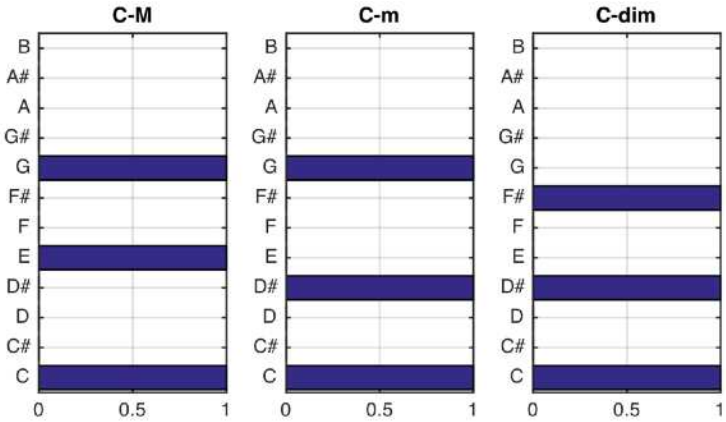
[T. Fujishima. "Realtime chord recognition of musical sound: a system using common lisp music". In Proc. of ICMC, 1999] LINK

Automatic Chord Recognition (ACR) Systems

(1) Frame-based/ template-based approach

- **Frame-based:**
  - chords are assigned to each temporal frame independently of each others
- **Template-based:**
  - chords are assigned by comparing chord-templates to chroma vectors
- **Chord templates**
  - 12 dimensional vector (if chroma has 12 dimensions) with value of
    - 1 if chroma exists in the chord
    - 0 if chroma does not exist in the chord
  - Template $G_a(c)$
    - chord-name, $a \in$ {C-M, C-m, C#-M, C#-m, ...}
    - chroma-index, $c \in [0,12[$
  - Variations [Gomez, 2006]
    - profiles can extended to the audio case (harmonics of each pitch) by considering a contribution of the h harmonic of each pitch with an amplitude of 0.6^{h-1}:

      use the first H=4 harmonics



[E. Gomez. Tonal description of polyphonic audio for mu- sic content processing. *INFORMS Journal on Computing*, 2006.]

Automatic Chord Recognition (ACR) Systems

(1) Frame-based/ template-based approach

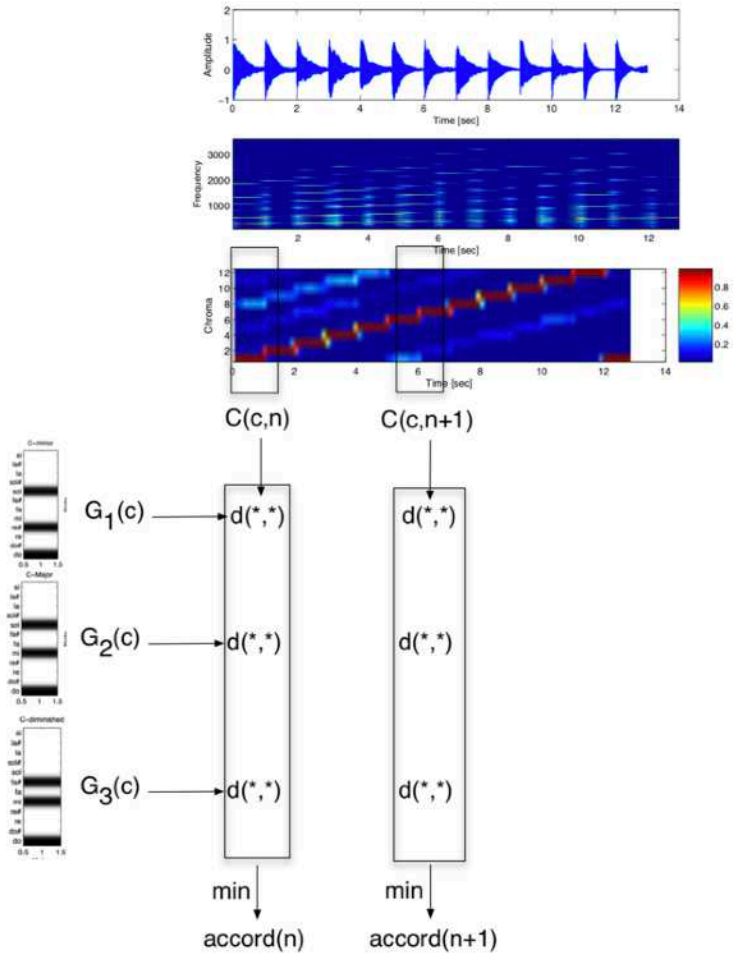
– **Frame-based estimation**

– **Inputs;**

- $C(c,n)$: chroma vector at time n
- $G_a(c)$ $a \in$ {C-M, C-m, C#-M, C#-m, ...}: set of chord -templates

– **Compute**

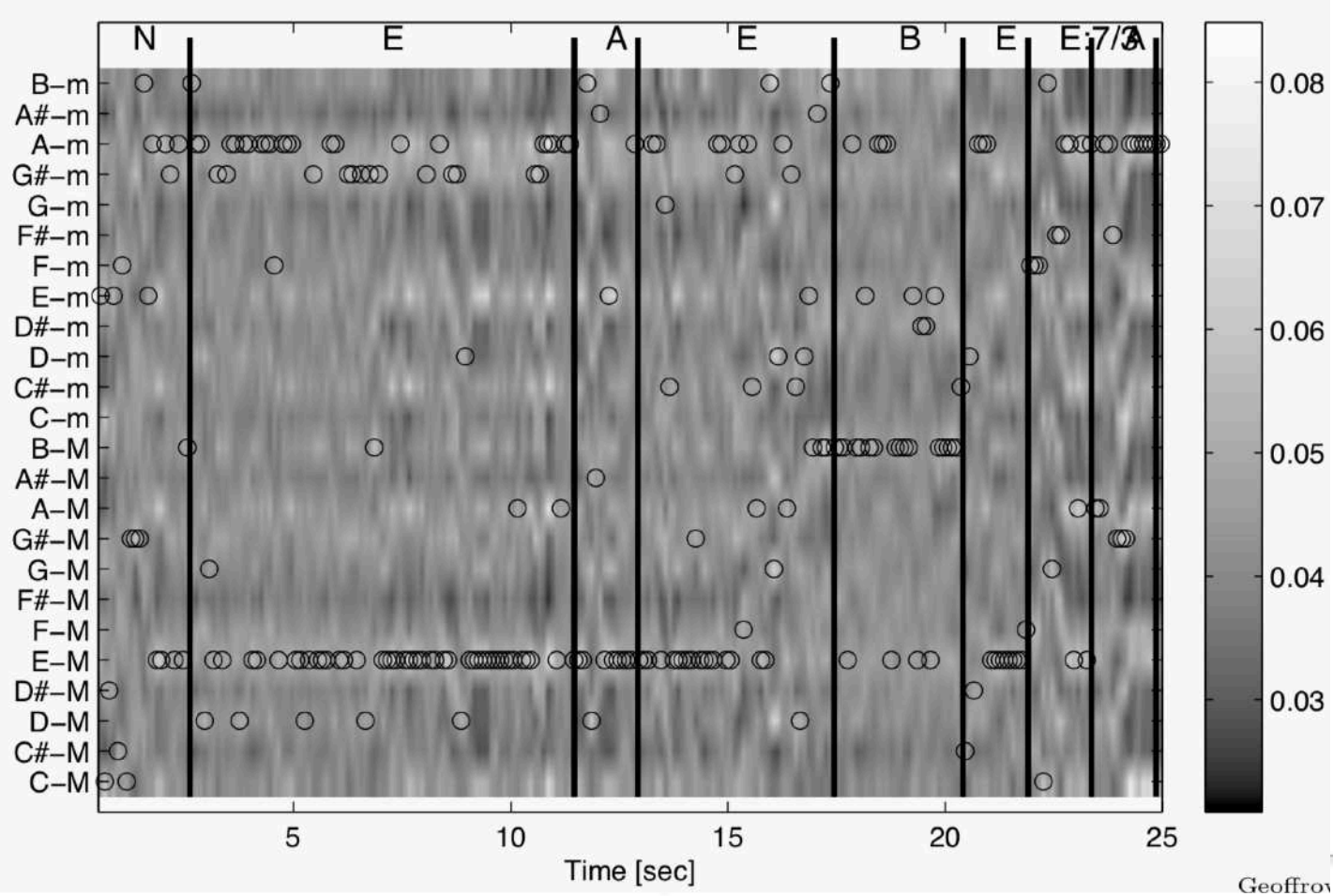
- $d(C(c,n), G_a(c))$ with d
  - Euclidean distance
  - cosine distance
- Choose the chord with the smallest distance
  - $\arg\min_a d(C(c,n), G_a(c))$



Automatic Chord Recognition (ACR) Systems

(1) Frame-based/ template-based approach

– **Result example**



Automatic Chord Recognition (ACR) Systems

(2) HMM-based approach

Chord Segmentation and Recognition using EM-Trained Hidden Markov Models

Alexander Sheh and Daniel P.W. Ellis
LabROSA, Dept. of Electrical Engineering,
Columbia University, New York NY 10027 USA
{asheh79,dpwe}@ee.columbia.edu

Abstract

Automatic extraction of content description from commercial audio recordings has a number of important applications, from indexing and retrieval through to novel musicological analyses based on very large corpora of recorded performances. Chord sequences are a description that captures much of the character of a piece in a compact form and using a modest lexicon. Chords also have the attractive property that a piece of music can (mostly) be segmented into time intervals that consist of a single chord, much as recorded speech can (mostly) be segmented into time intervals that correspond to specific words. In this work, we build a system for automatic chord transcription using speech recognition tools. For features we use "pitch class profile" vectors to emphasize the tonal content of the signal, and we show that these features far outperform cepstral coefficients for our task. Sequence recognition is accomplished with hidden Markov models (HMMs) directly analogous to subword models in a speech recognizer, and trained by the same Expectation-Maximization (EM) algorithm. Crucially, this allows us to use as input only the chord sequences for our training examples, without requiring the precise timings of the chord changes — which are determined automatically during training. Our results on a small set of 20 early Beatles songs show frame-level accuracy of around 75% on a forced-alignment task.

Keywords: audio, music, chords, HMM, EM.

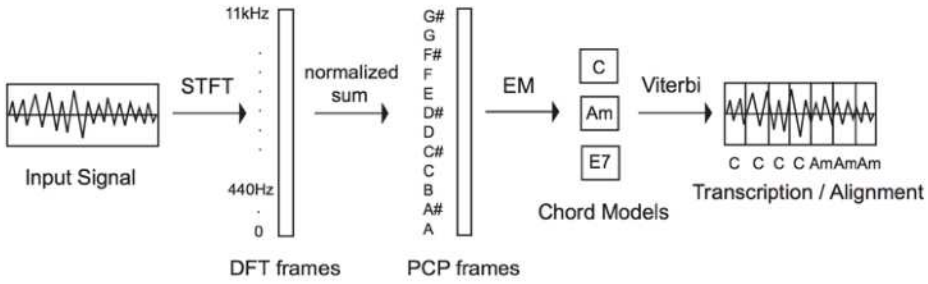
1 Introduction

The human auditory system is capable of extracting rich and meaningful data from complex audio signals. Machine listening research attempts to model this process using computers. In the music domain, there has been limited success when the input signal or analysis is relatively simple, i.e. single instrument, beat detection, etc. Unfortunately, for complex signals, such as ensemble performances, or more complex analyses, such as pitch transcription, the task rapidly increases in difficulty. In this paper we investigate a problem with complexity in both dimensions, chord recognition on unstructured, polyphonic, and multi-timbre audio. A system able to transcribe an arbitrary audio recording into an accurate chord sequence would have many applications in finding particular examples or themes in large audio databases, as well as enabling interesting new large-scale statistical analyses of musical content.

Our specific approach uses the hidden Markov models (HMMs) made popular in speech recognition (Gold and Morgan, 1999), including the sophisticated Expectation-Maximization (EM) algorithm used to train them. This is a statistical approach, in which the wide variety of feature frames falling under a single label is modeled as random variation that follows an estimated distribution. By making a direct analogy between the sequence of discrete, non-overlapping chord symbols used to describe a piece of music, and the word sequence used to describe recorded speech, much of the speech recognition framework can be used with minimal modification. In particular, no timing alignment is required between the chord labels and the training audio — using the constraints of the chord sequence alone, the EM approach converges to find optimal segmentations.

We drew on the prior work of Fujishima (1999) who proposed a representation of audio termed "pitch class profiles" (PCPs), in which the Fourier transform intensities are mapped to the twelve semitone pitch classes (chroma). This is very similar to the "chroma spectrum" proposed by Bartsch and Wakefield (2001). The assumption is that this representation captures harmonic information in a more meaningful way, thereby facilitating chord recognition. Fujishima's system uses nearest-neighbor classification to chord templates, and performed well on samples containing a single instrument.

Our system has parallels with the work by Raphael (2002), who also uses HMMs trained by EM to transcribe music in terms of chord labels. However, since his ultimate goal is note-level transcription, his "chord" vocabulary distinguishes between each different combination of simultaneous notes, in contrast to our approach of having a single model for "A minor" etc. This huge state space precludes direct training of models for each chord, and instead structural information about the harmonics expected for any given note combination are used to select among a relatively small set of model "factors", from which the desired



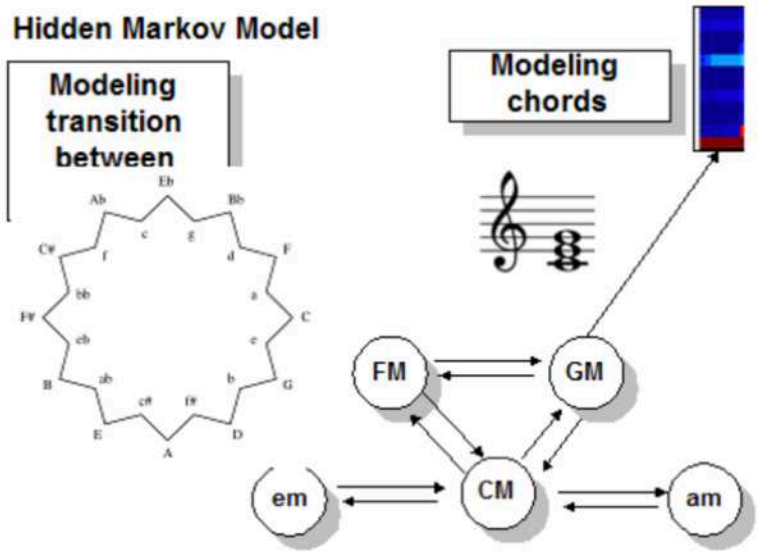
[A. Sheh and D. P. W. Ellis. "Chord segmentation and recognition using EM-trained hidden Markov models". In Proc. of ISMIR, 2003]

[H. Papadopoulos and G. Peeters. Large-scale study of chord estimation algorithms based on chroma representation. In Proc. of IEEE CBMI, 2007]

(2) HMM-based approach

– **Defining HMM for chord recognition**

- **Observations** O_t
  - sequence of chroma/PCP vectors over time t
- **States** $S_1, S_2, \ldots, S_N$
  - the chord labels (C-M, C-m, C#-M, ...)
- **Initial state distribution** $\pi = \{\pi_i\}$
  - set to uniform distribution (if we don't have informa
- **Emission probabilities** $b_j(O_t) = p(O_t \mid q_t = S_j)$
- **Transition probability** $a_{ij} = P(q_t = S_j \mid q_{t-1} = S_i)$



– **Goal**:

- Given a sequence of observations $O = O_1, O_2, \ldots, O_T$ and a model $\lambda = \{A, B, \pi\}$
  - find the most likely $Q = q_1, q_2, \ldots, q_T$ → **Viterbi decoding algorithm**

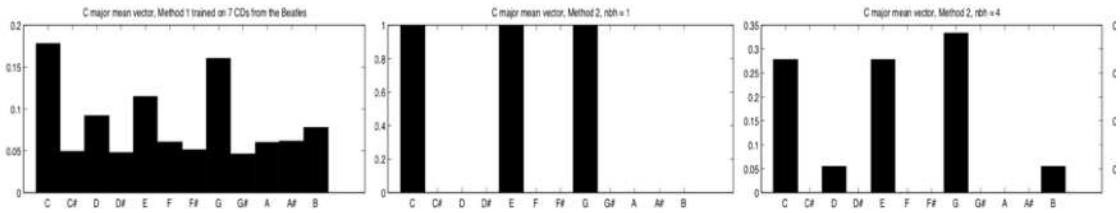
(2) HMM-based approach

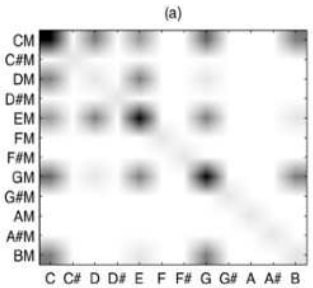
– **Emission probabilities** $b_j(O_t) = p(O_t \mid q_t = S_i)$

- can be **trained** on a corpus using Baum-Welch algorithm
- can be **based** on the (normalised) distance to chord-templates (see before)
- can be based on manually-tuned statistical models
  - multivariate Gaussian models with parameters μ and Σ
  - mean vectors and covariance matrices reflect musical knowledge

    mean vectors are 12-dim vectors with 1 if the note belongs to the chord

    considers the correlation between the chroma vectors corresponding to the pitch of the notes belonging to a given chord. In our method, we also consider the correlation between the harmonics of each note



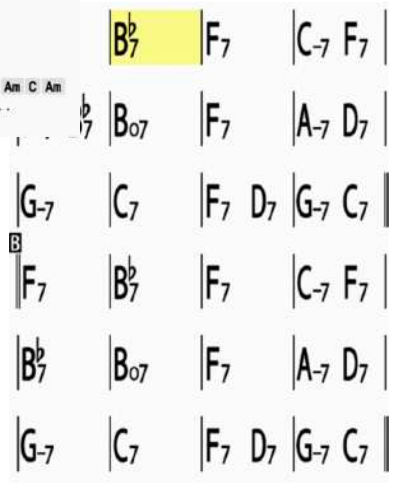


[H. Papadopoulos and G. Peeters. Large-scale study of chord estimation algorithms based on chroma representation. In Proc. of IEEE CBMI, 2007]

(2) HMM-based approach

- **Transition probability between chords**:
  - can be **trained** using the Baum-Welch algorithm
  - can be **trained** on a symbolic-music corpus
    - Guitar Tab, Real Book

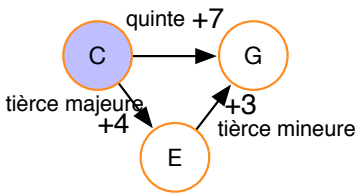
```
[Intro]
C Am C Am

[Verse 1]
  C                 Am
I heard there was a secret chord
     C                   Am
That David played and it pleased the lord
    F                G              C        G
But you don't really care for music, do you?
       C                 F              G
Well it goes like this the fourth, the fifth
    Am                  F
The minor fall and the major lift
    G             E7              Am
The baffled king composing hallelujah

[Chorus]
    F           Am          F           C    G   C      Am C Am
Hallelujah, hallelujah, hallelujah, hallelu-u-u-u-jah ....
```

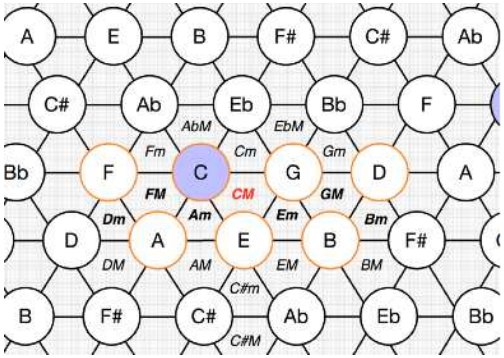


[Euler, Leonhard (1739). Tentamen novae theoriae musicae ex certissismis harmoniae principiis dilucide expositae (in Latin). Saint Petersburg Academy. p. 147.]

(2) HMM-based approach

- **Transition probability between chords**:
  - can be **based on musical rules**
    - distance between chords in Tonnetz space [Euler, 1739]





    - distance between chords in the Circle-of-Fitfh

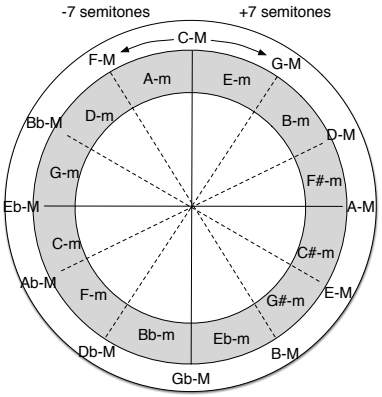
      sequence of fifths

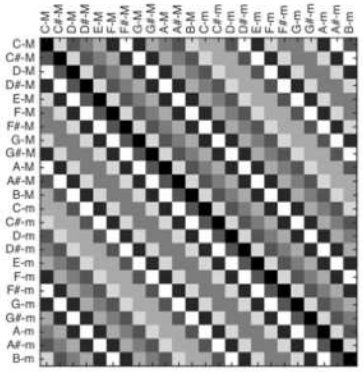
      relative Major-minor

      Example:

      G-M to C-M (consonance),

      G-M to Db-M (dissonance)



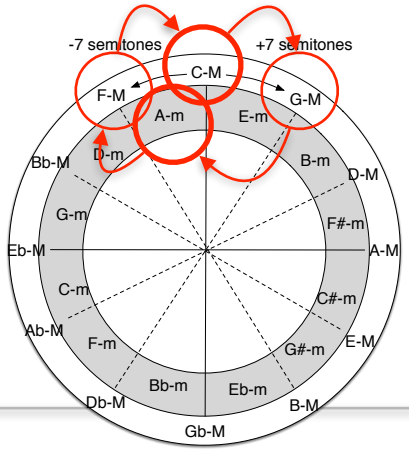


[Euler, Leonhard (1739). Tentamen novae theoriae musicae ex certissismis harmoniae principiis dilucide expositae (in Latin). Saint Petersburg Academy. p. 147.]

(2) HMM-based approach

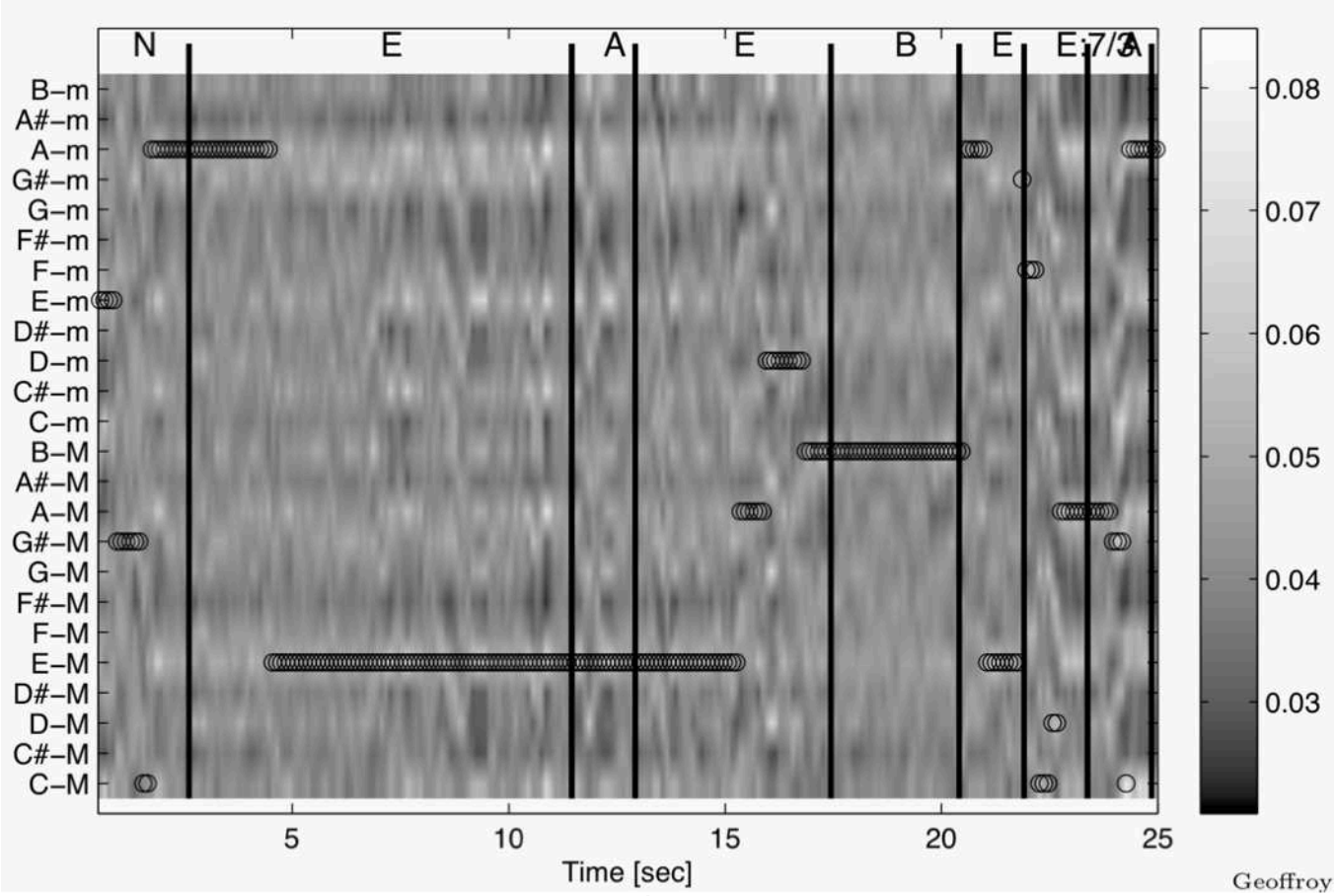
- **Transition probability between chords**:
  - can be **based on musical rules**
    - represent prototypical chord progressions

V → I:	G7 → Cmaj7
II → V → I:	D-7 → G7 → Cmaj7
Anatole (VI → II → V → I):	A-7 → D-7 → G7 → Cmaj7
12-bar Blues:	C7 →F7 →C7 →C7 ‖ F7 →F7 →C7 →C7 ‖ G7→F7 →C7 →G7 ‖
Magic 4-chords sequence in pop-music:	C → G → A- → F A- → F → C → G



(2) HMM-based approach

– **Result example**

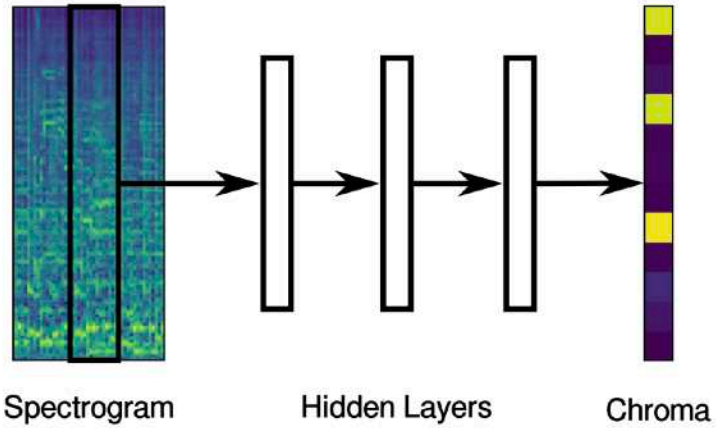


Automatic Chord Recognition (ACR) Systems

(2) Deep-learning-based approach A

– **Goal**:
  - standard Chroma extractors = too noisy features
  - replace the Chroma front-end by learned features
    – encode harmonic information important for chord recognition, while being robust to irrelevant interferences
    – train a 3-layers MLP to output a ground-truth chroma representation
    – ground-truth ? Chroma corresponding to the notes of the chord)
    – feeding the network with an audio spectrum with context instead of a single frame as input
  - **Deep Chroma**

– **Evaluation**
  - plug the output to a simple logistic regression to estimate the chord (no post-processing, smoothing)



	Btls	Iso	RWC	RW	Total
C	71.0±0.1	69.5 ±0.1	67.4±0.2	71.1±0.1	69.2±0.1
C^W_{Log}	76.0±0.1	74.2 ±0.1	70.3±0.3	74.4±0.2	73.0±0.1
S_{Log}	78.0±0.2	76.5 ±0.2	74.4±0.4	77.8±0.4	76.1±0.2
C_D	**80.2**±0.1	**79.3**±0.1	**77.3**±0.1	**80.1**±0.1	**78.8**±0.1

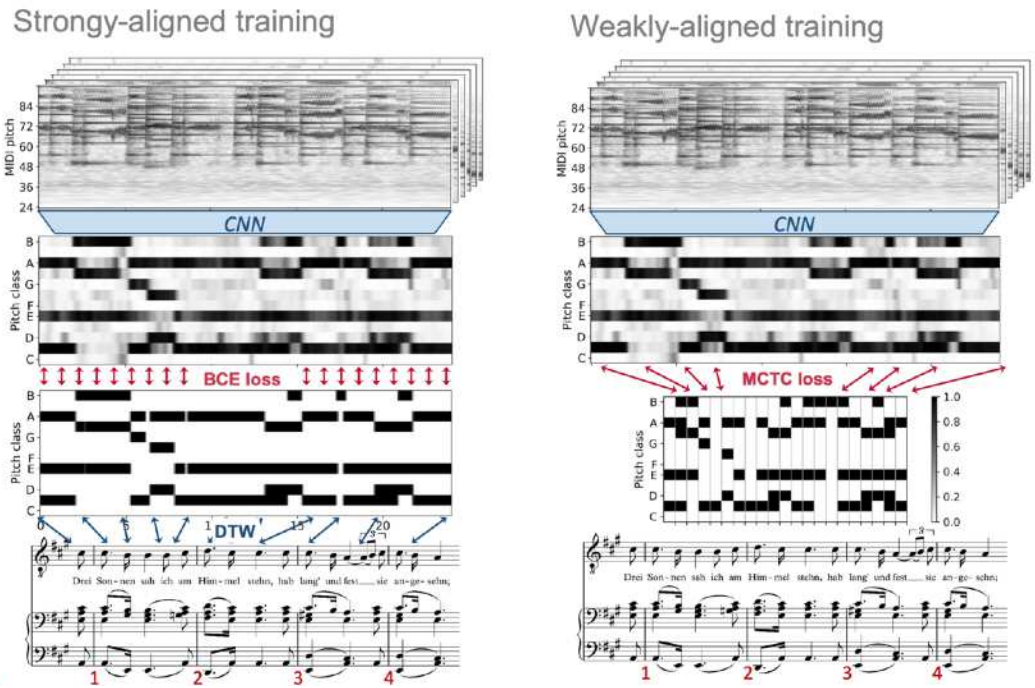
C: standard chroma from CQT
C^W_{Log}: chromagram with frequency weighting and logarithmic compression
S_{Log}: quarter-tone spectrogram
C_D: deep-chroma

[Filip Korzeniowski and Gerhard Widmer. "Feature learning for chord recognition: the deep chroma extractor". In ISMIR, 2016.] LINK

(2) Deep-learning-based approach A

– **Goal:**

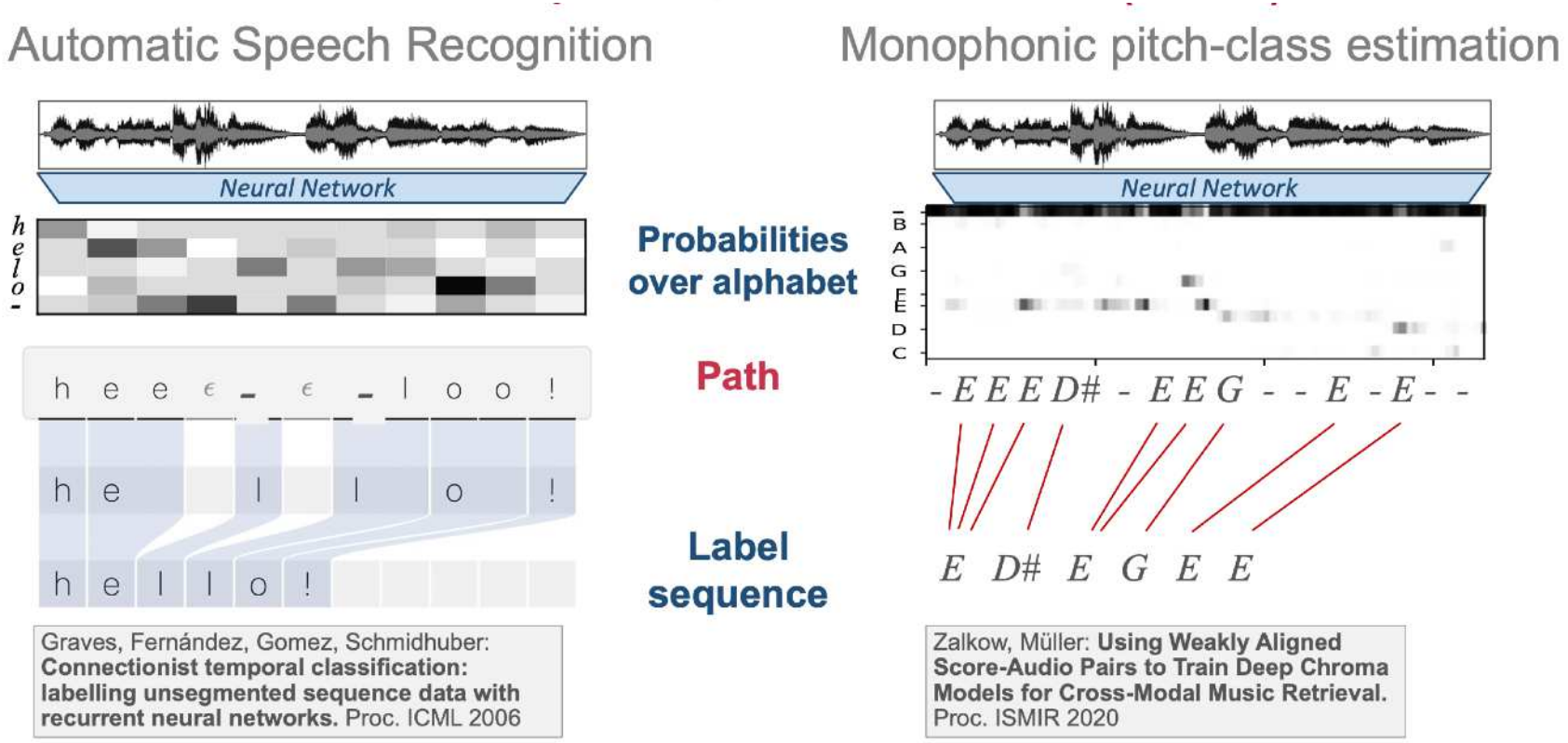
- replace the Chroma/PCP front-end by learned features
- Ground-truth ?
  - Aligned pitches (costly)
  - Non-aligned pitches (CTC)



[C. Weiß and G. Peeters. "Training deep pitch-class representations with a multi-label CTC loss". In Proc. of ISMIR, 2021] LINK

(2) Deep-learning-based approach A

– **Connectionist Temporal Classification (CTC) Loss**

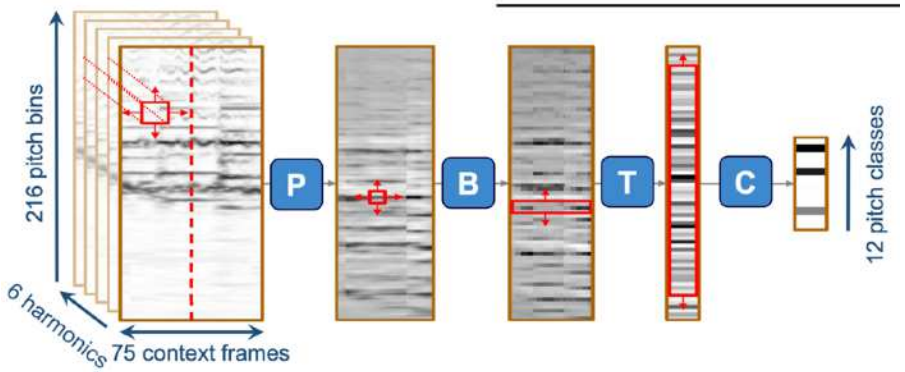


[C. Weiß and G. Peeters. "Training deep pitch-class representations with a multi-label CTC loss". In Proc. of ISMIR, 2021] LINK

(2) Deep-learning-based approach A

– CNN Architecture

- Input**: Harmonic-CQT**
- Simple **5-layerCNN**
- Roughly **48k parameters**
- **P**re-filtering, **B**inning to midi-pitches (216 →72), **T**emporal reduction (75 →1), **C**hroma reduction (72 →12)
- **Input**: Harmonic CQT



	Layer	Kernel size	Output shape	# Parameters
	Layer norm.		$(T+74, 216, 6)$	2592
P	Conv2D, MaxPool	15×15	$(T+74, 216, 20)$	27020
B	Conv2D, MaxPool	3×3	$(T+74, 72, 20)$	3620
T	Conv2D	75×1	$(T, 72, 10)$	15010
	Conv2D	1×1	$(T, 72, 1)$	11
C	Conv2D	1×61	$(T, 12+P, Q)$	$Q(62+73 \cdot P)$
	Total			48253 $+Q(62+73 \cdot P)$

[C. Weiß and G. Peeters. "Training deep pitch-class representations with a multi-label CTC loss". In Proc. of ISMIR, 2021] LINK

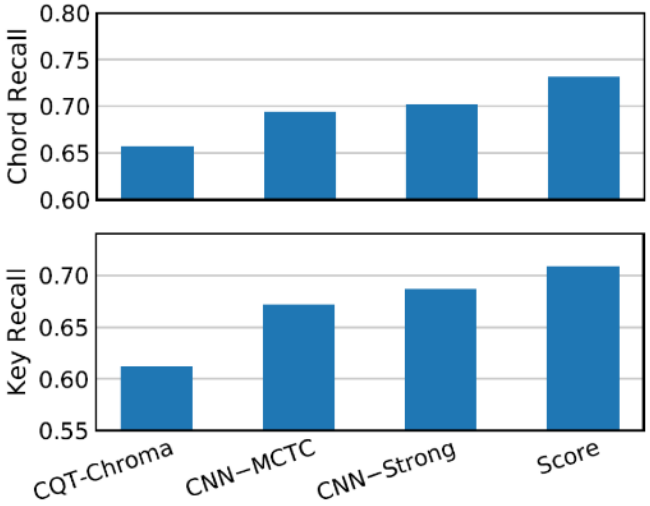
(2) Deep-learning-based approach A

– **Evaluation**

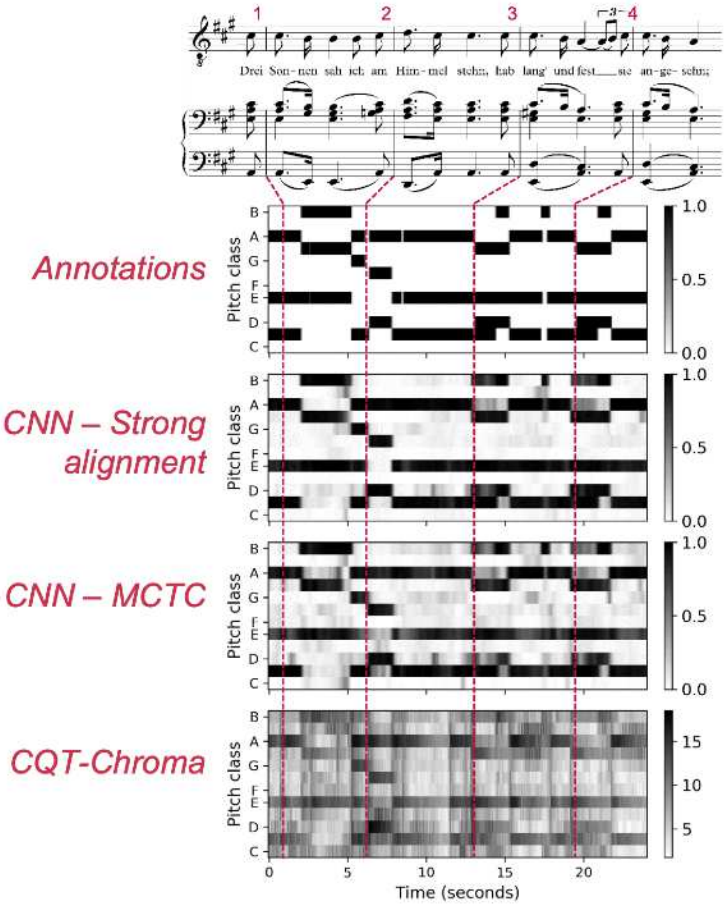
- Cosine similarity (CS), Average precision (AP)

Model/Loss	P	R	F	CS	AP
All-Zero	0	0	0	0.486	0.211
CQT-Chroma	0.512	0.681	0.579	0.701	0.594
CNN – SCTC	**0.850**	0.048	0.090	0.520	0.416
CNN – MCTC:NE	0.747	0.775	0.758	0.802	0.798
CNN – MCTC:WE	0.762	**0.853**	0.802	0.830	0.851
CNN – Strong alignment	**0.850**	0.790	**0.818**	**0.860**	**0.886**

– **Application**: Chord and Kev estimation



Application: Visualization



[C. Weiß and G. Peeters. "Training deep pitch-class representations with a multi-label CTC loss". In Proc. of ISMIR, 2021] LINK

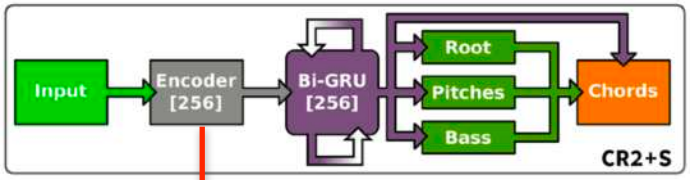
Automatic Chord Recognition (ACR) Systems

(2) Deep-learning-based approach B

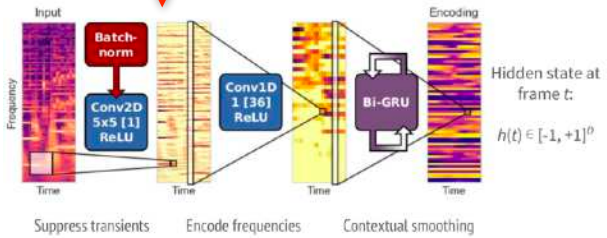
– **Goal 1:**
  - **End-to-end system**

– **Encoder:**
  - Input: T × F time-series of log-power constant-Q transform (CQT) spectra
  - First layer : can be interpreted as a harmonic saliency enhancer, as it tends to learn to suppress transients and vibrato while emphasizing sustained tones.
  - Second layer summarizes the pitch content of each frame, and can be interpreted as a local feature extractor

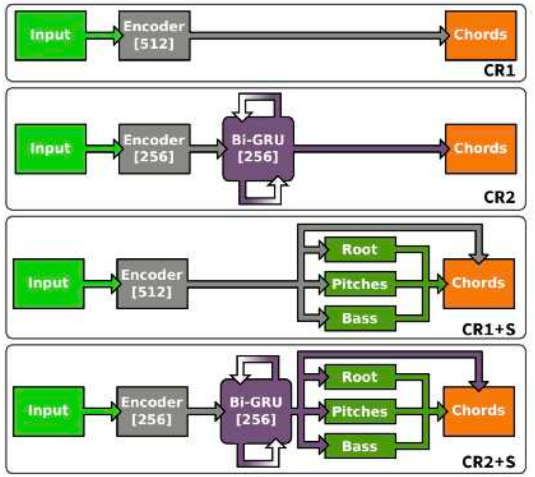
– **Decoder:**
  - 4 architectures



Encoder details



Decoder configurations



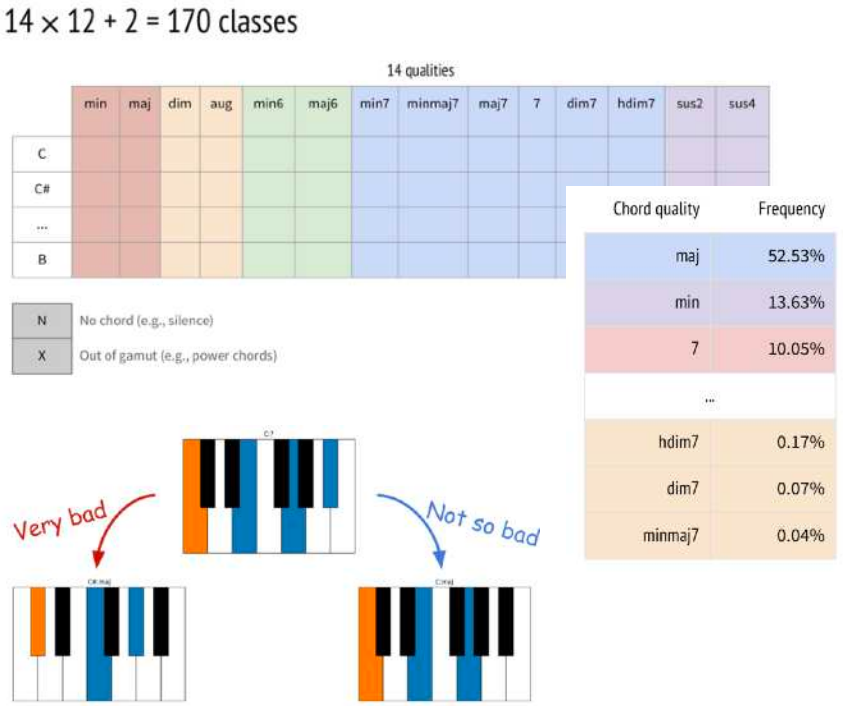
[B. McFee and J. P. Bello. "Structured training for large-vocabulary chord recognition". In Proc. of ISMIR, 2017] LINK

(2) Deep-learning-based approach B

- **Goal 1:**
  - **End-to-end system**
- **Goal 2:**
  - **Large chord vocabularies**
    - Classes are not well-separated

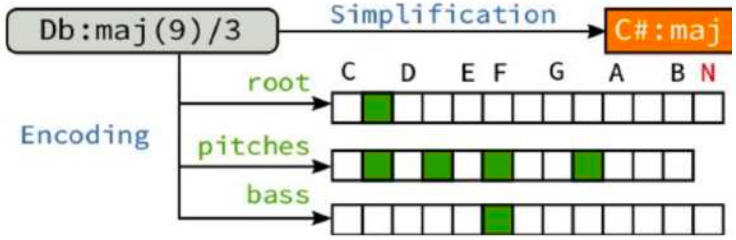
      C:7 = C:maj + m7

      C:sus4 vs. F:sus2
    - Class distribution is non-uniform
    - Rare classes are hard to model
  - Take into account the fact that **some mistakes are better than others**

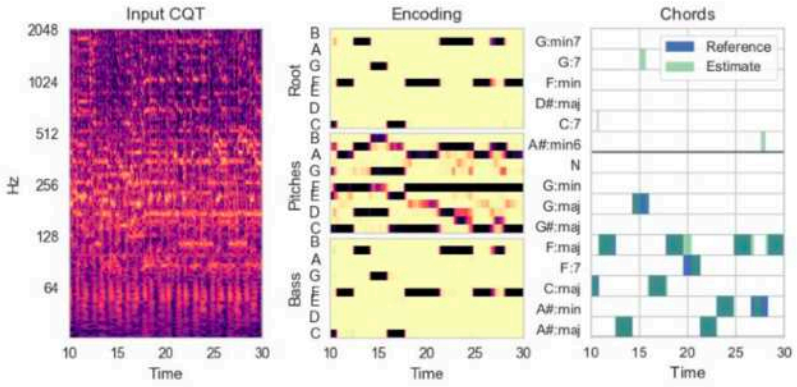


[B. McFee and J. P. Bello. "Structured training for large-vocabulary chord recognition". In Proc. of ISMIR, 2017] LINK

(2) Deep-learning-based approach B



- **Idea**:
  - Exploit the implicit chord space structure
- Represent chord labels as **binary encodings**
  - Similar chords with different labels will have similar encodings
  - Dissimilar chords will have dissimilar encodings
- **Learning problem:**
  - Predict the **encoding** from audio
  - Learn to decode into chord **labels**
- **Model architecture**
  - Input: constant-Q spectral patches
  - Per-frame outputs:
    - Root [multi-class, C=13]
    - Pitches [multi-label, C+12]
    - Bass [multi-class, C=13]
    - Chords [multi-class, C=170]
  - Convolutional-recurrent architecture (encoder-decoder)
  - End-to-end training



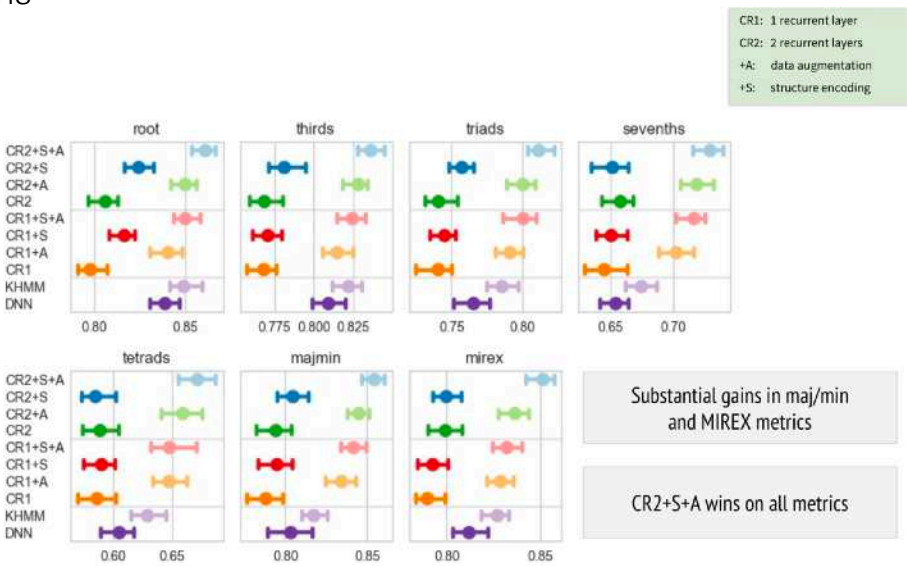
[B. McFee and J. P. Bello. "Structured training for large-vocabulary chord recognition". In Proc. of ISMIR, 2017] LINK

Automatic Chord Recognition (ACR) Systems

(2) Deep-learning-based approach B

- **What about root bias?**
  - Quality and root should be independent
  - But the data is inherently biased
- **Solution: data augmentation**!
  - Pitch-shift the audio and annotations simultaneously
  - Each training track → ± 6 semitone shifts
    - All qualities are observed in all root positions
    - All roots, pitches, and bass values are obs
- **8 configurations**
  - ± data augmentation
  - ± structured training
  - vs. 2 recurrent layers
- **1217 recordings**
  - (Billboard + Isophonics + MARL corpus)
  - 5-fold cross-validation
- **Baseline models:**
  - DNN [Humphrey & Bello, 2015]
  - KHMM [Cho, 2014]

- **Training details**
  - Keras / TensorFlow + pescador
  - ADAM optimizer
  - Early stopping @20, learning rate reduction @10
    - Determined by decoder loss
  - 8 seconds per patch
  - 32 patches ber batch
  - 1024 batches per epoch



[B. McFee and J. P. Bello. "Structured training for large-vocabulary chord recognition". In Proc. of ISMIR, 2017] LINK

Automatic Chord Recognition (ACR)
Evaluation

Automatic Chord Recognition (ACR)
Evaluation

Task definition:

- https://www.music-ir.org/mirex/wiki/2019:Audio_Chord_Estimation

- Given an audio track
  → estimate the set of temporal (start:end) segments and associated chord label

```
0.000000 2.612267 N
2.612267 11.459070 E
11.459070 12.921927 A
12.921927 17.443474 E
17.443474 20.410362 B
20.410362 21.908049 E
21.908049 23.370907 E:7/3
23.370907 24.856984 A
24.856984 26.343061 A:min/b3
26.343061 27.840748 E
27.840748 29.350045 B
29.350045 35.305963 E
35.305963 36.803650 A
36.803650 41.263102 E
41.263102 44.245646 B
44.245646 45.720113 E
45.720113 47.206190 E:7/3
47.206190 48.692267 A
48.692267 50.155124 A:min/b3
50.155124 51.652811 E
51.652811 53.138888 B
53.138888 56.111043 E
56.111043 65.131995 A
65.131995 68.150589 B
68.150589 71.192403 A
71.192403 74.199387 E
74.199387 75.697074 A
75.697074 80.236575 E
80.236575 83.208730 B
83.208730 86.221693 E
86.221693 87.736621 A
87.736621 89.257528 A:min/b3
89.257528 90.720385 E
```

Automatic Chord Recognition (ACR)
Evaluation

Datasets:

- https://www.audiocontentanalysis.org/data-sets/

- QMUL Isophonics (Beatles, Carole King, Queen, Michael Jackson)
  - http://isophonics.net/datasets
- AIST RWC (Real World Computing)
  - https://staff.aist.go.jp/m.goto/RWC-MDB/
- McGill Billboard
  - https://ddmal.music.mcgill.ca/research/The_McGill_Billboard_Project_(Chord_Analysis_Dataset)/
- GuitarSet
  - https://guitarset.weebly.com/
- ...

Performance measures

– **Two criteria to evaluate**
  - **(1)** get the correct segment boundaries (independently of the labels)
  - **(2)** get the correct label at each time

Automatic Chord Recognition (ACR) Evaluation

Example of results:

Submissions

	Abstract	Contributors
CM1	PDF	Chris Cannam, Matthias Mauch
JLCX1, JLCX2	PDF	Junyan Jiang, Ke Chen, Wei Li, Guangyu Xia
SG1	PDF	Franz Strasser, Stefan Gaser
FK2	[PDF]	Florian Krebs, Filip Korzeniowski, Sebastian Böck

Results

Summary

All figures can be interpreted as percentages and range from 0 (worst) to 100 (best).

Isophonics2009

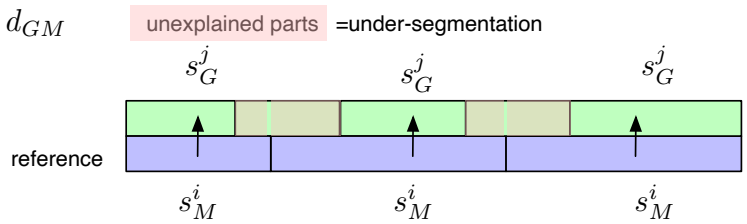
Algorithm	MirexRoot	MirexMajMin	MirexMajMinBass	MirexSevenths	MirexSeventhsBass	MeanSeg	UnderSeg	OverSeg
CM1	78.66	75.51	72.58	54.78	52.36	85.87	87.22	85.98
FK2	87.38	86.80	83.43	75.55	72.60	89.29	87.24	92.35
JLCX1	86.75	86.25	84.44	75.87	74.39	90.33	88.36	93.38
JLCX2	86.51	86.05	84.23	75.64	74.17	90.12	87.94	93.48
SG1	82.03	78.67	76.84	69.20	67.55	82.34	89.03	77.87

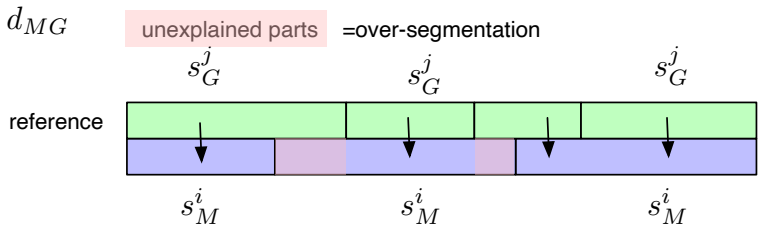
Automatic Chord Recognition (ACR) Evaluation

(1) get the correct segment boundaries (independently of the labels)

- Notation
  - ground-truth segments: S_G^j
  - estimated (measured) segments: S_M^i
- directional Hamming distance d_{GM}:
  - For each S_M^i find the segment S_G^j with the maximum overlap → then summing the differences
    - $d_{GM} = \sum_{S_M^i} \sum_{S_G^k \neq S_G^j} |S_M^i \cap S_G^k|$

      where $|.|$ denotes the length of a segment
    - d_{GM} indicates missed boundaries (**under-segmentation**)
- inverse-directional Hamming distance d_{MG}:
  - For each S_G^j find the segment S_M^i with ...
    - d_{MG} indicates segment fragmentation (**over-segmentation**)
- **Chord segmentation**: $Q = 1 - \frac{\text{max. of } d_{GM} d_{MG}}{\text{total duration of song}}$





[Abdallah, Samer A., Katy Noland, Mark B. Sandler, Michael Casey, and Christophe Rhodes."Theory and Evaluation of a Bayesian Music Structure Extractor." In Proc. of ISMIR, 2005]

Automatic Chord Recognition (ACR)
Evaluation

(2) get the correct label at each time

_ **Chord symbol recall (CSR)**: $CSR = \frac{\text{total duration of segments where annotation equal estimation}}{\text{total duration of annotated segments}}$

– **Need to choose a chord vocabularies**

- Chord root-note only
- Major and minor: `{N, maj, min}`
- Seventh chords: `{N, maj, min, maj7, min7, 7}`
- Major and minor with inversions: `{N, maj, min, maj/3, min/b3, maj/5, min/5}`
- Seventh chords with inversions: `{N, maj, min, maj7, min7, 7, maj/3, min/b3, maj7/3, min7/b3, 7/3, maj/5, min/5, maj7/5, min7/5, 7/5, maj7/7, min7/b7, 7/b7}`

– **Vocabulary mapping**

- `G:7(#9)` is mapped to `G:maj` because the interval set of `G:maj, {1,3,5}`, is a subset of the interval set of the `G:7(#9), {1,3,5,b7,#9}`
- `G:7(#9)` is mapped to `G:7` instead because the interval set of `G:7 {1, 3, 5, b7}`
- non-possible mapping (excluded from evaluation)
  - `D:aug` or `F:sus4(9)` to `{maj, min}`

Hidden Markov Model (HMM)

Hidden Markov Model (HMM)

- Markov model
  - Andreï A. Markov (1856-1922): Russian mathematician

- Markov chain:
  - a stochastic process with discrete time t, each time can be in one of the possible discrete states $S_{i\in\{1,\ldots,N\}}$
  - q_t is the value of the state at time t

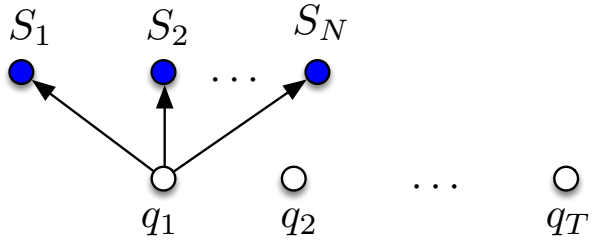
- First order Markov chain:
  - the prediction of the current state q_t depends only on the previous time q_{t-1}

$$p(q_t \mid q_{t-1}, q_{t-2} \cdots q_0) = p(q_t \mid q_{t-1})$$

source: https://fr.wikipedia.org/wiki/Andre%C3%AF_Markov_(math%C3%A9maticien)

Observed Markov model

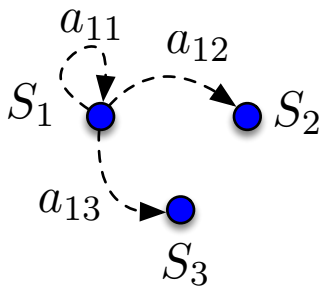
- The system is in one of a set of N distinct **states** $S_1, S_2, \ldots, S_N$
- We denote the **time** instants as $t = 1, 2, \ldots, T$
- We denote the **actual state** at time t as q_t
- We denote the sequence of actual state $Q = q_1, q_2, \ldots, q_T$



- **Transition probability**
  - Discrete first order Markov chain
    - $P(q_t = S_j \mid q_{t-1} = S_i, q_{t-2} = S_k, \ldots) = P(q_t = S_j \mid q_{t-1} = S_i)$
  - is independent of time
    - $a_{ij} = P(q_t = S_j \mid q_{t-1} = S_i) \quad 1 \leq i, j \leq N$

      with $a_{ij} \geq 0$

      $$\sum_j a_{ij} = 1$$



- **Initial state distribution** $\pi = \{\pi_i\}$ with
  - $\pi_i = P(q_1 = S_i)$

Observed Markov model example

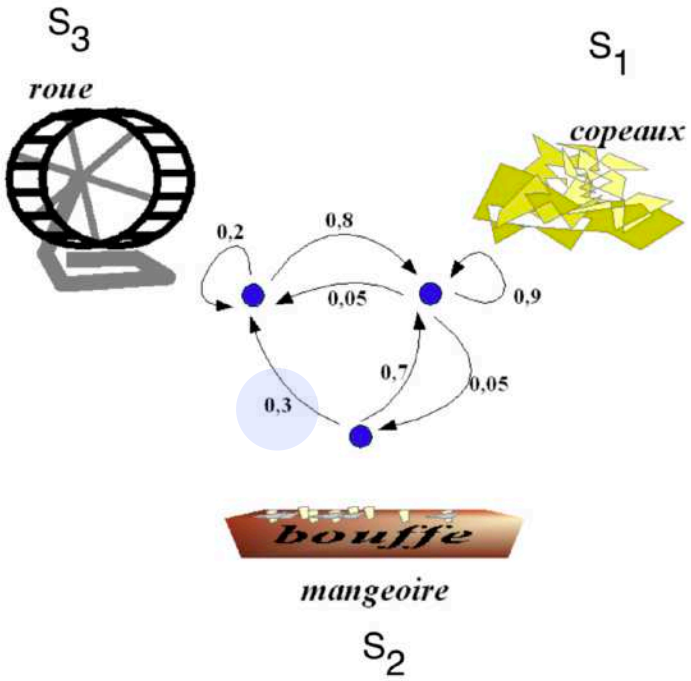
- Doudou the hamster has 3 states during a day
  - he sleeps:
    - he is in state S_1 (sawdust shavings)
  - he eats:
    - he is in state S_2 (feeder)
  - he does exercice:
    - he is in state S_3 (hamster wheel)

- We can represent the succession of states a transition matrix $A = (a_{ij})$ between states

$$A = \begin{pmatrix} 0.9 & 0.05 & 0.05 \\ 0.7^{(i=2,j=1)} & 0 & 0.3 \\ 0.8 & 0 & 0.2 \end{pmatrix}$$

(i = row, j = column)

$$\sum_j a_{ij} = 1$$



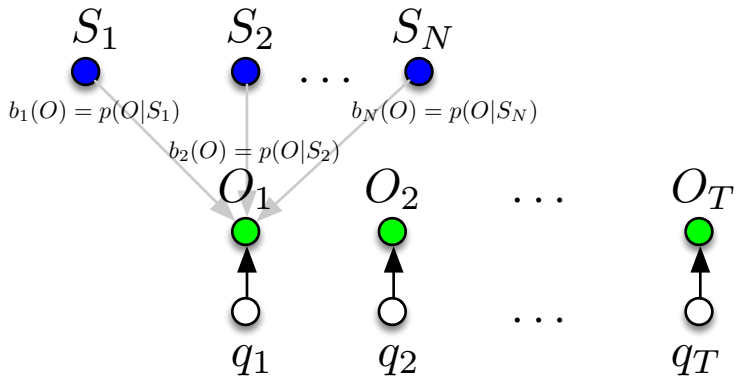
Modèle de Markov d'une journée de Doudou le hamster

source: https://fr.wikipedia.org/wiki/Cha%C3%AEne_de_Markov

Hidden Markov model

– We do not observe directly the actual states $Q = q_1, q_2, \ldots, q_T$ but an emission of those
  - a sequence of **observations** $O = O_1, O_2, \ldots, O_T$

– **Emission probability**
  - The observation is a probabilistic function of the state S_j
  - $b_j(O) = P(O \mid S_j)$



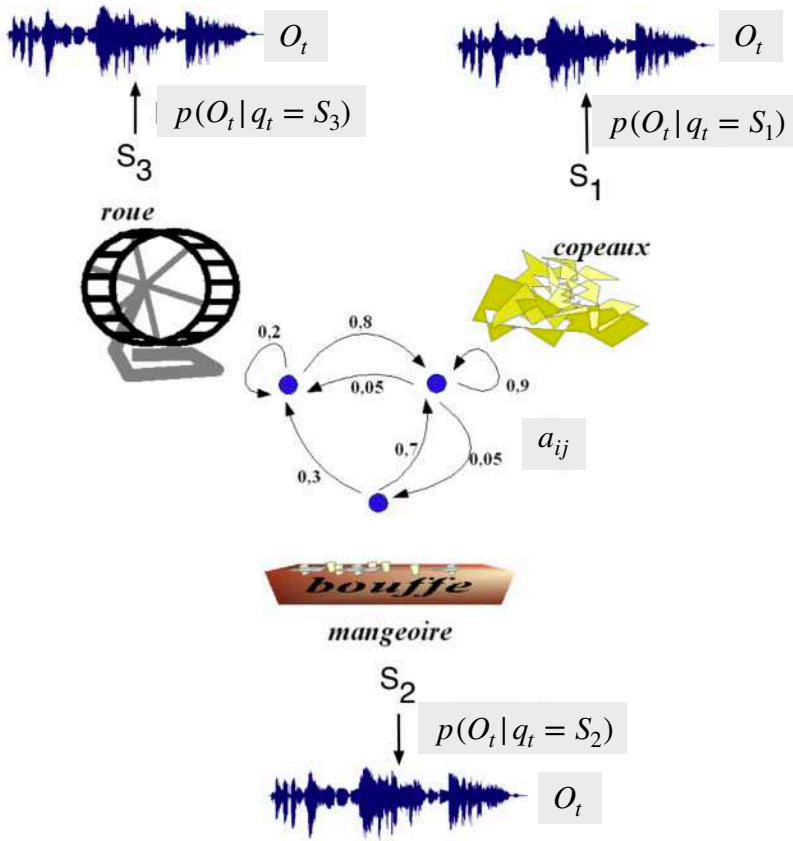
  - **Discrete Observations:** set of symbols v_k
    – $B = \{ b_j(k)\}$ with $b_j(k) = P(O_t = v_k \mid q_t = S_j)$
    the probability of observing symbol k in state j

  - **Continuous Observation Densities:** Gaussian Mixture Model
    – $b_j(O) = \sum_m c_{j(m)} \mathcal{N}(O \mid \mu_{j(m)}, \Sigma_{j(m)})$

– We denote by $\lambda = \{A, B, \pi\}$ the set of elements defining an HMM

Hidden Markov model example

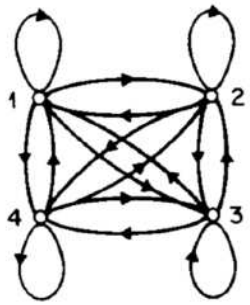
- In a hidden Markov model, we do not observe directly the states $Q = q_1, q_2, \ldots, Q_T$,
  - they are "hidden"
- we observe an emission O_t of the states q_t
  - example: we observe the sound $O = O_1, O_2, \ldots, O_T$ made by Doudou the hamster
- For each state, we can define
  - an emission probability given state S_j: $b_j(O_t) = p(O_t \mid q_t = S_j)$



state: chord
observation: chroma

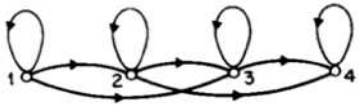
Various topologies of Markov models

– Ergodic/Fully Connected HMMs
  - a HMM allowing for transitions from any emitting state to any other emitting state



$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

– Left-Right HMMs
  - an HMM where the transitions are not allowed to states which indices are lower than the current state: $a_{ij} = 0, \quad j < i$

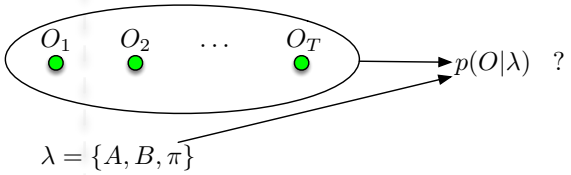


$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 \\ 0 & a_{22} & a_{23} & a_{24} \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{bmatrix}.$$

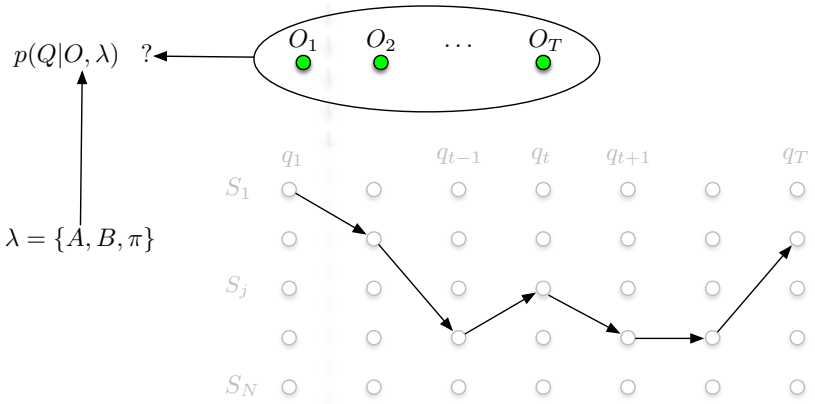
source: L. Rabiner, 1989.

Hidden Markov Model (HMM)
The three basic problems for HMMs

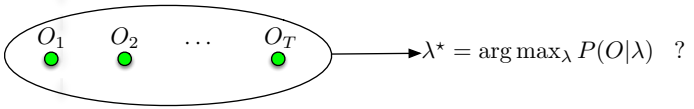
Problem 1. Likelihood of a model (Forward algorithm)



Problem 2. Decoding of the best sequence (Viterbi algorithm)

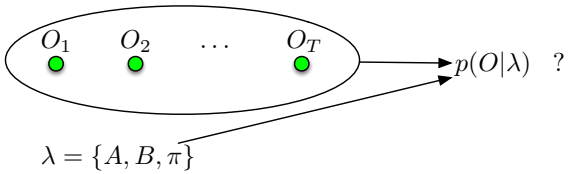


Problem 3. Training a HMM model (Forward-Backward, Baum-Welch algorithm)



[L. Rabiner. "A tutorial on hidden markov model and selected applications in speech". Proceedings of the IEEE, 77(2):257–285, 1989.] LINK

Problem 1. Likelihood of a model (Forward algorithm)



- Given
  - a sequence of observations $O = O_1, O_2, \dots, O_T$
  - a model $\lambda = \{A, B, \pi\}$
- What is the probability that the model has generated O ?
  - $P(O \mid \lambda)$?
- Application example:
  - does the observation sequence O corresponds to
    - the model λ_1 {sleep/eat/exercice} of Doudou the hamster ?
    - the model λ_2 {sleep/metro/work} of Bill the employee ?
- **Problem:**
  - We don't know which states $Q = q_1, q_2, \dots, q_T$ the model has gone through
    - We need to consider all possibilities

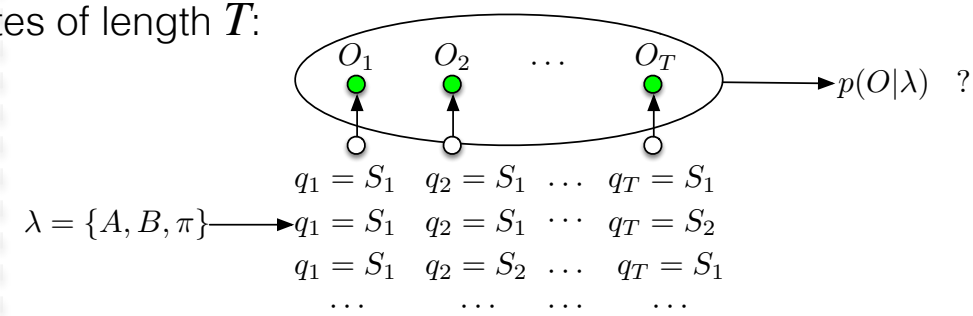
[L. Rabiner. "A tutorial on hidden markov model and selected applications in speech". Proceedings of the IEEE, 77(2):257–285, 1989.] LINK

Problem 1. Likelihood of a model (Forward algorithm)

Method 1 (exhaustive)

– We need to enumerate all sequences of states of length T:

$$Q = q_1, q_2, \ldots, q_T$$



– For each sequence Q, we estimate

- The probability of the observations O

$$P(O \mid Q, \lambda) = \Pi_{t=1}^{T} P(O_t \mid q_t, \lambda) = b_{q_1}(O_1) \cdot b_{q_2}(O_2) \ \ldots \ b_{q_T}(O_T)$$

- The probability of this specific sequence Q

$$P(Q \mid \lambda) = \pi_{q_1} \cdot a_{q_1 q_2} \cdot a_{q_2 q_3} \ \ldots \ a_{q_{T-1} q_T}$$

- The joint probability of O and Q

$$P(O, Q \mid \lambda) = P(O \mid Q, \lambda)\, P(Q \mid \lambda) = \pi_{q_1} \cdot b_{q_1}(O_1) \cdot a_{q_1, q_2} \ \ldots \ a_{q_{T-1}, q_T} \cdot b_{q_T}(O_T)$$

– Finally, we need to sump up for all possible sequences Q

$$P(O \mid \lambda) = \sum_{\forall Q} P(O, Q \mid \lambda) = \sum_{\forall q_1, q_2, \ldots} P(O \mid Q, \lambda)\, P(Q \mid \lambda)$$

– Huge computational cost !!!

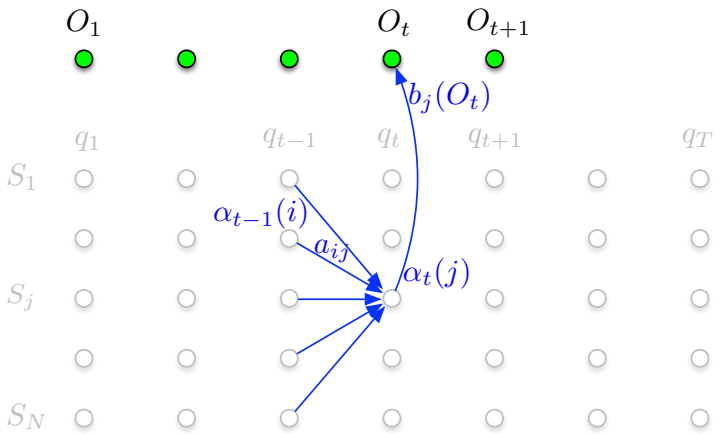
Problem 1. Likelihood of a model (Forward algorithm)

Method 2 (optimised): Forward algorithm

– **Definition**: **Forward variable**

$$\alpha_t(j) \stackrel{def}{=} P(O_1, O_2, \ldots, O_t, q_t = S_j \,|\, \lambda)$$

- probability of the partial observation sequence $O_1, O_2, \ldots, O_t$ (until time t) and "state S_j at time t" given the model λ



- Initialisation

$$\alpha_1(i) = \pi_i \cdot b_i(O_1)$$

- Loop

$$\alpha_t(j) = \left[\sum_{i=1}^{N} \alpha_{t-1}(i) \cdot a_{ij} \right] b_j(O_t)$$

- Ending

$$P(O \,|\, \lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

Proof:

$$\alpha_T(i) = P(O_1, O_2, \ldots, O_T, q_t = S_i \,|\, \lambda)$$

$$\sum_i \alpha_T(i) = \sum_i P(O_1, O_2, \ldots, O_T, q_t = S_i \,|\, \lambda)$$

$$= P(O_1, O_2, \ldots, O_T \,|\, \lambda)$$

Problem 1. Likelihood of a model (Forward algorithm)

Method 2 (optimised): revisited

– We could also have used the reverse recursion

– **Definition: Backward variable**

$$\beta_t(i) \stackrel{def}{=} P(O_{t+1}, O_{t+2}, \ldots, O_T | q_t = S_i, \lambda)$$

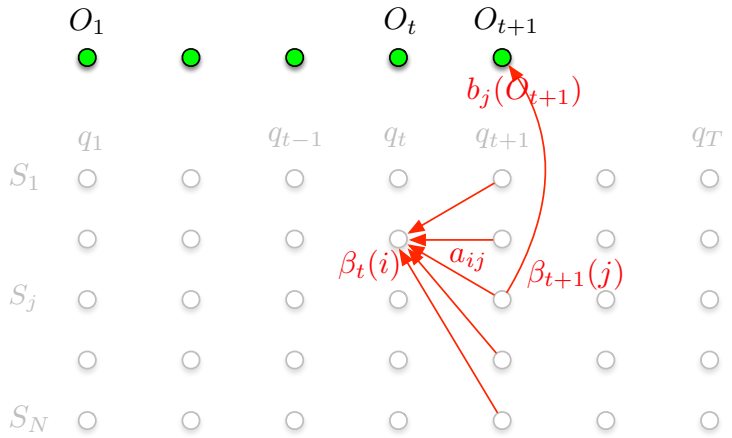
- probability of the partial observation sequence from $t+1$ to T given "state S_i at time t", and the model λ

- Initialisation

$$\beta_T(i) = 1$$

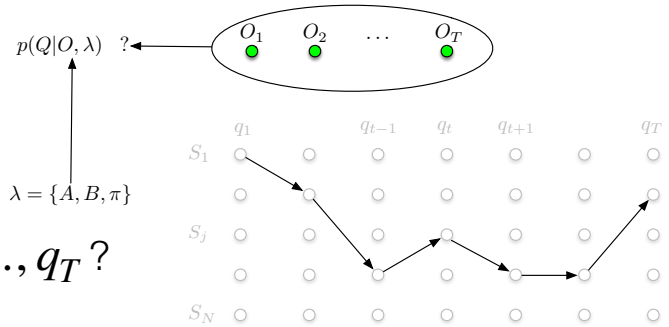
- Loop

$$\beta_t(i) = \sum_j a_{ij} \cdot b_j(O_{t+1}) \cdot \beta_{t+1}(j)$$



Problem 2. Decoding of the best sequence (Viterbi algorithm)

- Given
  - a sequence of observations $O = O_1, O_2, \ldots, O_T$
  - a model $\lambda = \{A, B, \pi\}$



- What is the corresponding sequence of states $Q = q_1, q_2, \ldots, q_T$?
  - $Q^\star = \arg\max_Q P(Q \mid O, \lambda)$?

- Application example:
  - if we observe the sequence of sounds O produced by Doudou and given its model λ {sleep/eat/exercice}, what is the corresponding sequence of activities Q of Doudou ?

- **Problem**:
  - Find the most **optimal** sequence Q
    - Several possible definitions for **optimality**

[L. Rabiner. "A tutorial on hidden markov model and selected applications in speech". Proceedings of the IEEE, 77(2):257–285, 1989.] LINK

Problem 2. Decoding of the best sequence (Viterbi algorithm)

Optimality-1: choose the q_t which are individually the most likely

– **Définition**

$$\gamma_t(i) \stackrel{def}{=} P(q_t = S_i \,|\, O, \lambda)$$

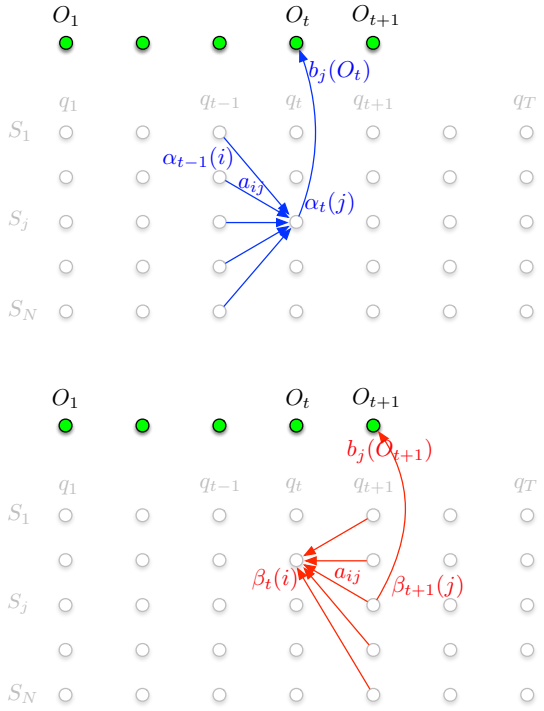
- probability of "being in state S_i at time t" given the observation sequence O and the model λ

- using $p(q \,|\, O, \lambda)\; p(O \,|\, \lambda) = p(q, O \,|\, \lambda)$

Reminder:

$$\alpha_t(i) = P(O_1, O_2, \ldots, O_t, q_t = S_i \,|\, \lambda)$$
$$\beta_t(i) = P(O_{t+1}, O_{t+2}, \ldots, O_T \,|\, q_t = S_i, \lambda)$$

$$\gamma_t(i) = \frac{p(q_t = S_i, O \,|\, \lambda)}{p(O \,|\, \lambda)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_q p(q, O \,|\, \lambda)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_i \alpha_t(i)\beta_t(i)}$$



_ We choose $q_t = \arg\max_i [\gamma_t(i)]$

- Problem: it does not take into account the transition probabilities (what about if $a_{ij} = 0$?)

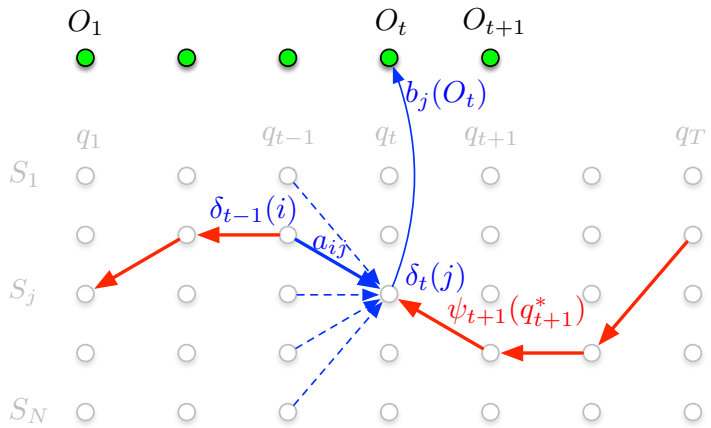
Problem 2. Decoding of the best sequence (Viterbi algorithm)

Optimality-2: choose the Q which is globally the most likely

– **Dynamic Programming**

- $\delta_t(j) \stackrel{def}{=} \max_{q_1,q_2,\ldots,q_{t-1}} P(O_1, O_2 \ldots O_t, q_1, q_2 \ldots q_t = S_j | \lambda)$
  - it is the highest probability of the partial alignment starting at 1 and ending at t in state S_j
- We can compute $\delta_t(*)$ using $\delta_{t-1}(*)$ as

$$\delta_t(j) = \max_i \left[\delta_{t-1}(i)\, a_{ij}\right]\; b_j(O_t)$$



– **Viterbi algorithm**

- **Forward pass**
  - Initialisation:

$$\delta_1(i) = \pi_i\; b_i(O_1)$$

  - Loop

    for t=2:T and for j=1:N

$$\delta_t(j) = \max_i \left(\delta_{t-1}(i)\, a_{ij}\right)\; b_j(O_t)$$

  - Ending

$$P^* = \max_i \delta_T(i)$$

- **Backward pass**
  - for T=T-1,T-2,...,1

$$q_t^* = \psi_{t+1}(q_{t+1}^*)$$

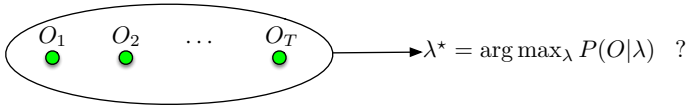
and $\psi_1(i) = 0$

and $\psi_t(j) = \arg\max_i \left(\delta_{t-1}(i)\, a_{ij}\right)$

and $q_T^* = \arg\max_i \delta_T(i)$

Problem 3. Training a HMM model (Forward-Backward, Baum-Welch algorithm)

- Given
  - a sequence of observations $O = O_1, O_2, \ldots, O_T$



- What is the model $\lambda^\star$ that maximises the likelihood of the observations:
  - $\lambda^\star = \arg\max_\lambda P(O \mid \lambda)$
  - i.e. find the parameters $\lambda = \{A, B, \pi\}$ that maximises the likelihood of the observation's sequence O

- Application example:
  - find the parameters of the model λ of Doudou the hamster ?

[L. Rabiner. "A tutorial on hidden markov model and selected applications in speech". Proceedings of the IEEE, 77(2):257–285, 1989.] LINK

Problem 3. Training a HMM model (Forward-Backward, Baum-Welch algorithm)

Baum-Welch algorithm

– **EM (Expectation-Maximisation) algorithm**

– **Definition**

$$\xi(i,j) \stackrel{def}{=} P(q_t = S_i, q_{t+1} = S_j \,|\, O, \lambda)$$

- using $p(q \,|\, O, \lambda)\, p(O \,|\, \lambda) = p(q, O \,|\, \lambda)$

$$\xi(i,j) = \frac{P(q_t = S_i, q_{t+1} = S_j, O \,|\, \lambda)}{P(O \,|\, \lambda)}$$

$$= \frac{\alpha_t(i)\; a_{ij}\; b_j(O_{t+1})\; \beta_{t+1}(j)}{\sum_i \sum_j \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}$$

– We also have

$$\gamma_t(i) = \sum_j \xi_t(i,j)$$

Reminder:

$$\alpha_t(i) = P(O_1, O_2, \ldots, O_t, q_t = S_i \,|\, \lambda)$$
$$\beta_t(i) = P(O_{t+1}, O_{t+2}, \ldots, O_T \,|\, q_t = S_i, \lambda)$$
$$\gamma_t(i) = P(q_t = S_i \,|\, O, \lambda)$$

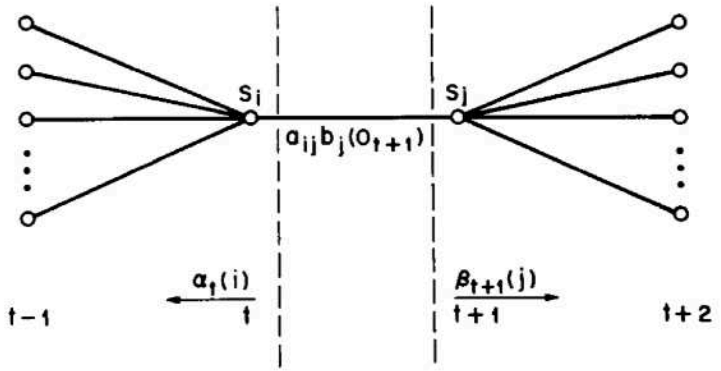


Fig. 6. Illustration of the sequence of operations required for the computation of the joint event that the system is in state S_i at time t and state S_j at time $t + 1$.

source: L. Rabiner, 1989.

Problem 3. Training a HMM model (Forward-Backward, Baum-Welch algorithm)

Baum-Welch algorithm (cont.)

- $\sum_{t=1}^{T-1} \gamma_t(i)$ = expected (over time) number of times that S_i is visited
- $\sum_{t=1}^{T-1} \xi_t(i,j)$ = expected (over time) number of transitions from S_i to S_j

– Re-estimation of the HMM parameters

- π_i = expected frequency (number of times) in S_i at time $(t = 1)$ = $\gamma_1(i)$
- $a_{ij} = \dfrac{\text{expected number of transitions from state to state } S_i \text{ to state } S_j}{\text{expected number of transitions from state } S_i} = \dfrac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$

– a) Discrete observations

- $b_j(k) = \dfrac{\text{expected number of times in state } S_j \text{ an observing symbol } v_k}{\text{expected number of times in state } S_j} = \dfrac{\sum_{t=1 s.t. O_t=v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$

Problem 3. Training a HMM model (Forward-Backward, Baum-Welch algorithm)

Baum-Welch algorithm (cont.)

– **a) Discrete observations**

- $b_j(k) = \dfrac{\text{expected number of times in state } S_j \text{ an observing symbol } v_k}{\text{expected number of times in state } S_j} = \dfrac{\sum_{t=1 s.t. O_t=v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$

Baum-Welch algorithm (cont.)

– b) Continuous Observation Densities in HMMs

- Gaussian Mixture Model with $m \in \{1,\ldots,M\}$ components for state S_j

$$b_j(O) = \sum_{m=1}^{M} c_{jm} \cdot \mathcal{N}(O, \mu_{jm}, \Sigma_{jm})$$

- $\gamma_t(j,k)$ is the probability of being in state j at time t with the k^{th} mixture component

$$\gamma_t(j,k) = \left[\frac{\alpha_t(j)\beta_t(j)}{\sum_{j=1}^{N}\alpha_t(j)\beta_t(j)}\right] \cdot \left[\frac{c_{jk}\mathcal{N}(O_t,\mu_{jk},\Sigma_{jk})}{\sum_{m=1}^{M} c_{jm}\mathcal{N}(O_t,\mu_{jm},\Sigma_{jm})}\right]$$

- Re-estimating HMM parameters for state j and k^{th} mixture component
  - mixture coefficients: $c_{jk} = \frac{\sum_{t=1}^{T}\gamma_t(j,k)}{\sum_{t=1}^{T}\sum_{k=1}^{M}\gamma_t(j,k)}$
  - mean vectors: $\mu_{jk} = \frac{\sum_{t=1}^{T}\gamma_t(j,k)\cdot O_t}{\sum_{t=1}^{T}\gamma_t(j,k)}$
  - covariance matrice: $\Sigma_{jk} = \frac{\sum_{t=1}^{T}\gamma_t(j,k)\cdot(O_t-\mu_{jm})(O_t-\mu_{jm})'}{\sum_{t=1}^{T}\gamma_t(j,k)}$

Problem 3. Training a HMM model (Forward-Backward, Baum-Welch algorithm)

Baum-Welch algorithm (cont.)

– **HMM parameters constrains**

$$\sum_{i=1}^{N} \pi_i = 1$$

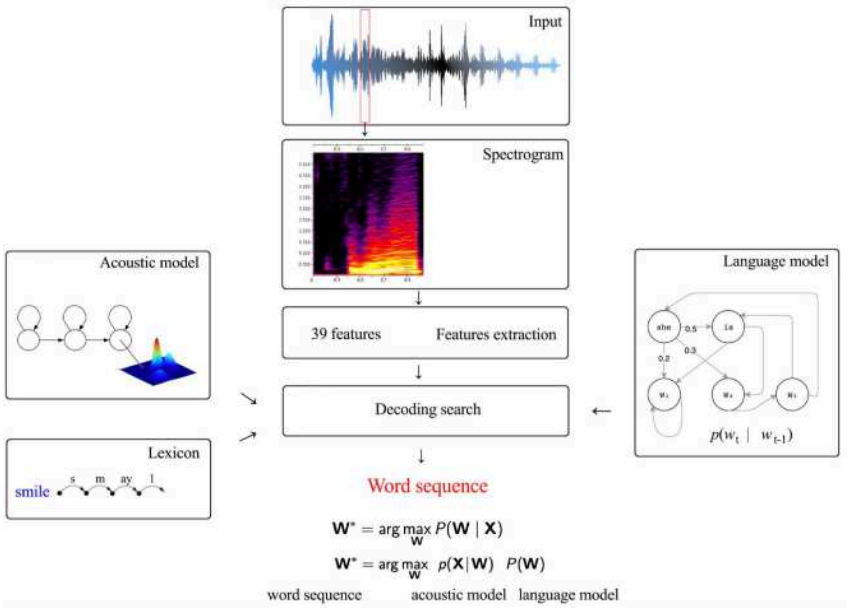
$$\sum_{j=1}^{N} a_{ij} = 1, \quad 1 \leq i \leq N$$

$$\sum_{k=1}^{M} b_j(k) = 1, \quad 1 \leq j \leq N$$

Automatic Speech Recognition (ASR) using GMM/HMM

Automatic Speech Recognition using GMM/HMM

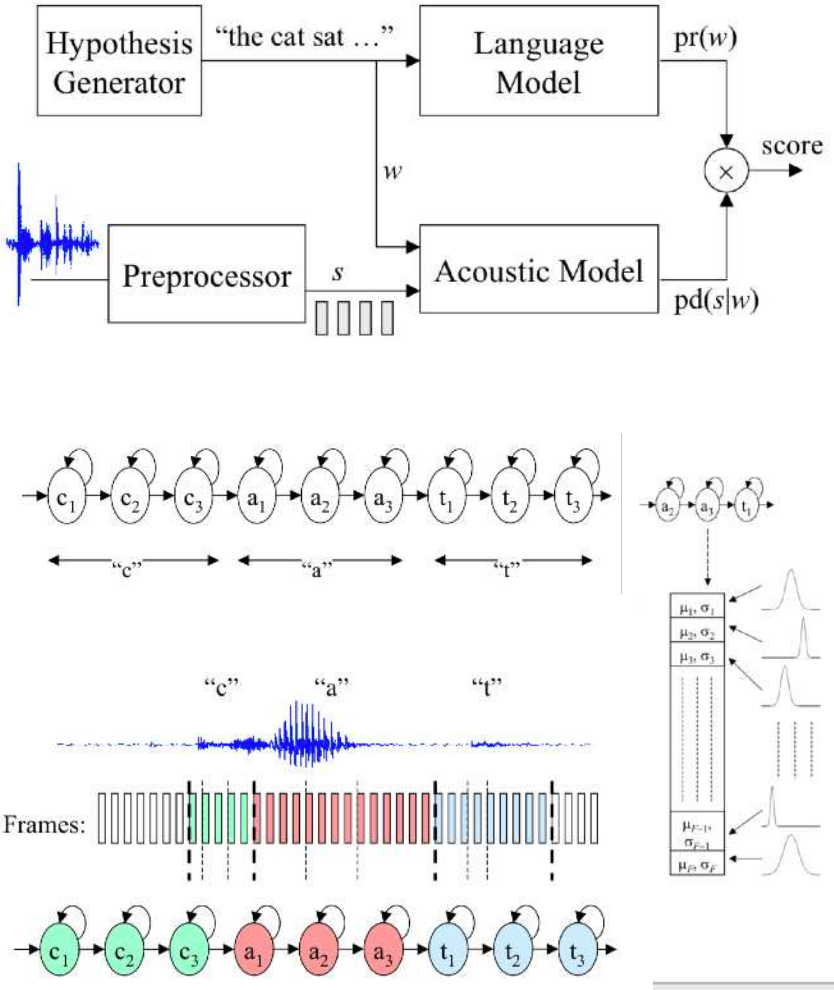
- Given a speech signal s
- Find the most likely word sequence
  - $p(W \mid X) = \dfrac{p(X \mid W) \cdot p(W)}{p(X)}$
- $p(W)$:
  - prior probability of the word sequence w obtained from a **Language model**
- $p(X \mid W)$
  - likelihood of the word sequence w obtained from stored models of speech (**Acoustic model**)



source: https://jonathan-hui.medium.com/speech-recognition-gmm-hmm-8bb5eff8b196

Automatic Speech Recognition using GMM/HMM

- **Isolated Word Recognition**
  - each phoneme in a word corresponds to a number of model states (typically 3 states per phoneme)
  - when saying a word, the speaker stays in each state for one or more frames and then goes on to the next state
  - the emission probability of each state is modeled as a GMM over the MFCC/Δ/Δ^2
  - $p(s \mid w)$ is obtained by Viterbi alignment
  - choose the word with the highest probability
  - separate HMM for each word in the vocabulary
- **Continuous Speech Recognition**



source: Brooke, 2002, Speech Processing

What you need to know

Lecture What you need to know

- What is a Source/Filter model
- What are MFCCs ?
- What are the Chroma/ Pitch-Class-Profile
  - how they are computed
  - what are there advantages/ drawbacks
- What is the Constant-Q-Transform
  - what problem does it solve?
  - how does it solve it ?
- What is a pitch
  - what is a temperament (tuning-system)
- What is a Hidden Markov Model
  - what are the variables
  - what is the Viterbi decoding algorithm
- How are chords estimated using an HMM approach, a DL approach ?
- How are the performances of such a system measured ?