MIRtoolbox: Sound and music analysis of audio recordings using Matlab

MUS4831, Olivier Lartillot, 26.10.2017

Part 1

- *MIRtoolbox* overview
- Basic signal processing operators
- Auditory models
- Pitch estimation
- Timbral descriptions

Part 2 (in 2 weeks)

- Rhythm, metrical structure
- Tonal analysis
- Segmentation, structure
- Statistical descriptions, similarity
- Music & emotion
- Advanced use

Lecture-Workshop

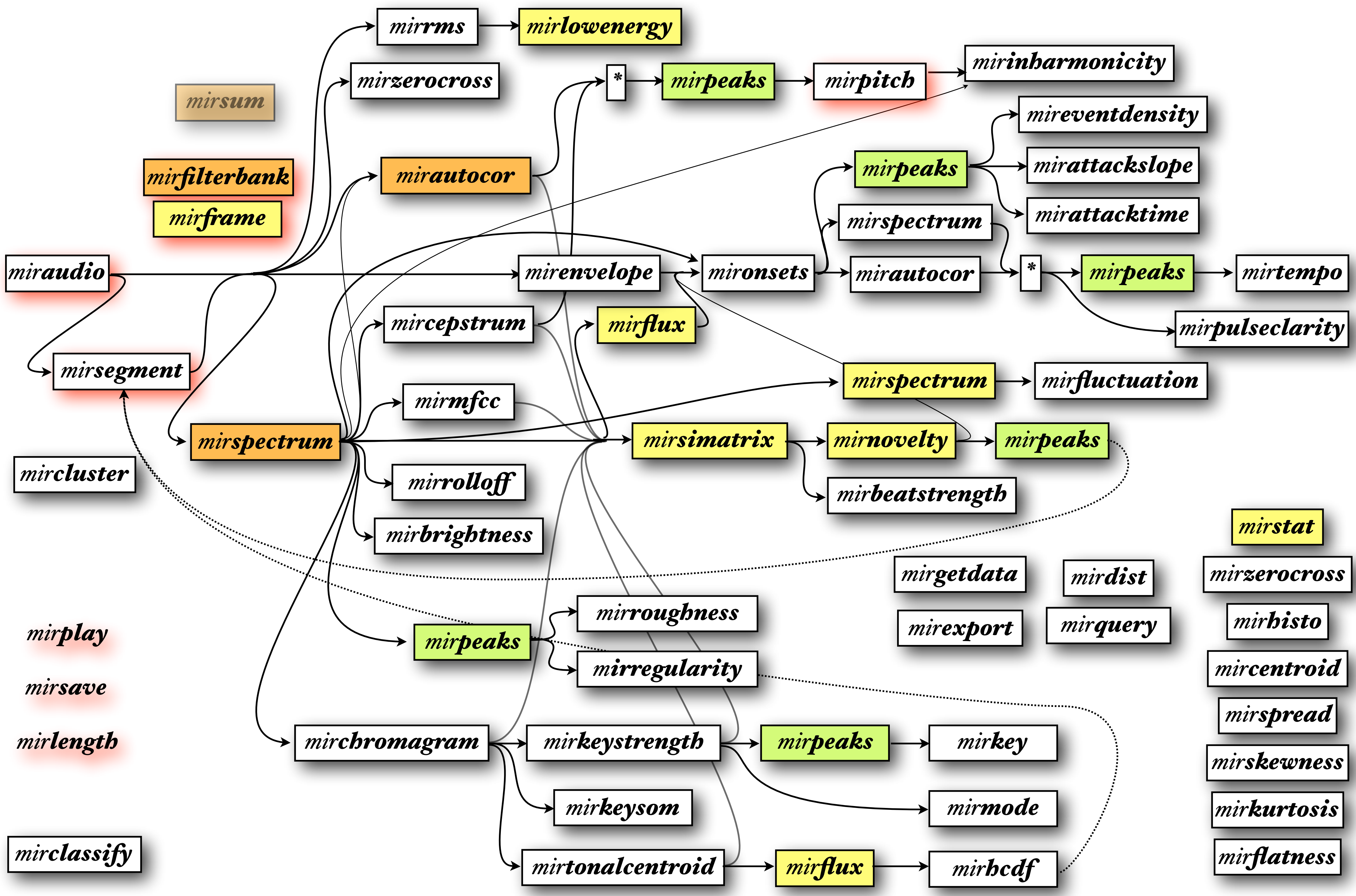
- Lecture slides in PDF
- Workshop handout
- We will install *MIRtoolbox* together…
- Sound examples
- Useful: *MIRtoolbox* User's Guide

1. Overview & syntax

General Principles

- Why did we create *MIRtoolbox*?
  - Research project about **music & emotion**
  - Analysis tool for students from various background
- Modular framework: Building blocks
- Simple and adaptive syntax
  - User can focus on the general design.
  - *MIRtoolbox* takes care of the technical details.
- Free software, open source
- One standard tool for MIR study and research (10000s downloads)

MIRtoolbox Features



Let's now install *MIRtoolbox*…

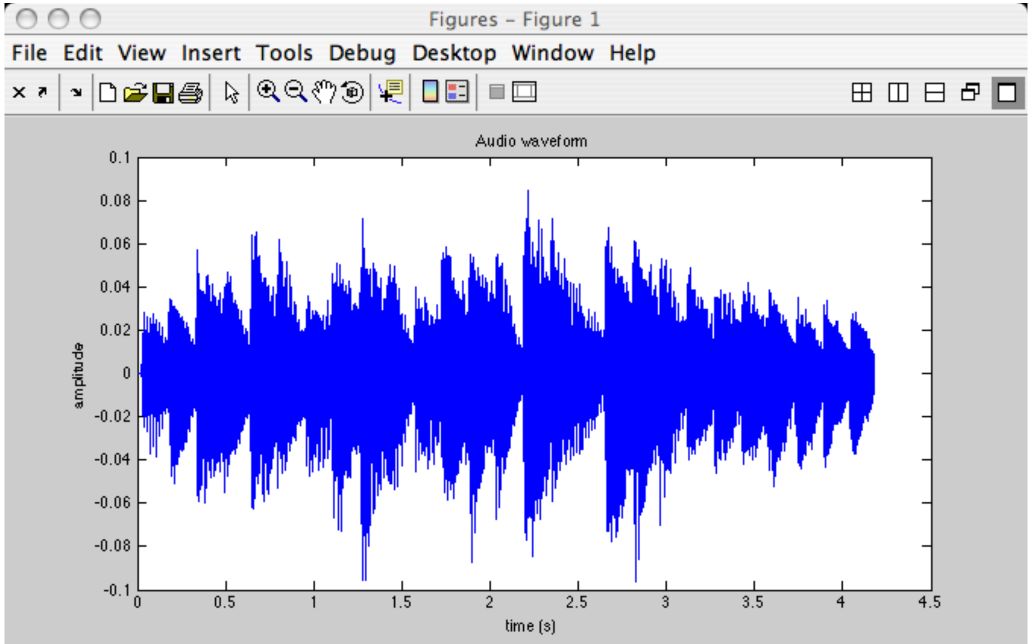
http://bit.ly/mirtoolbox

Requires:

- *Matlab*,
- *Signal Processing* toolbox,
- *Statistics and Machine Learning* toolbox

Basic Operations

miraudio('ragtime.wav')



.wav
.mp3
.mp4
.m4a
.ogg
.flac
.au

*miraudio('**Folder**')*

'***Folder***' = all files in Current Directory

miraudio(..., 'Extract')

extraction options

- *miraudio(..., '**Extract**', 1, 2)*
  extracts signal from 1 s to 2 s after the start

```
a = miraudio('ragtime.wav')
b = miraudio(a, 'Extract', 1,2)
b = miraudio('ragtime.wav', 'Extract', 1,2)
mirplay(b)
mirsave(b)
```

miraudio(..., 'Trim')

trimming options

- *miraudio('ragtime.wav', '**Trim**')*

  trims (pseudo-)silence at start and end

  - *miraudio(..., '**TrimStart**')* at start only
  - *miraudio(..., '**TrimEnd**')* at end only

- *miraudio(..., '**TrimThreshold**', t)*

  specifies the silence threshold t = .06

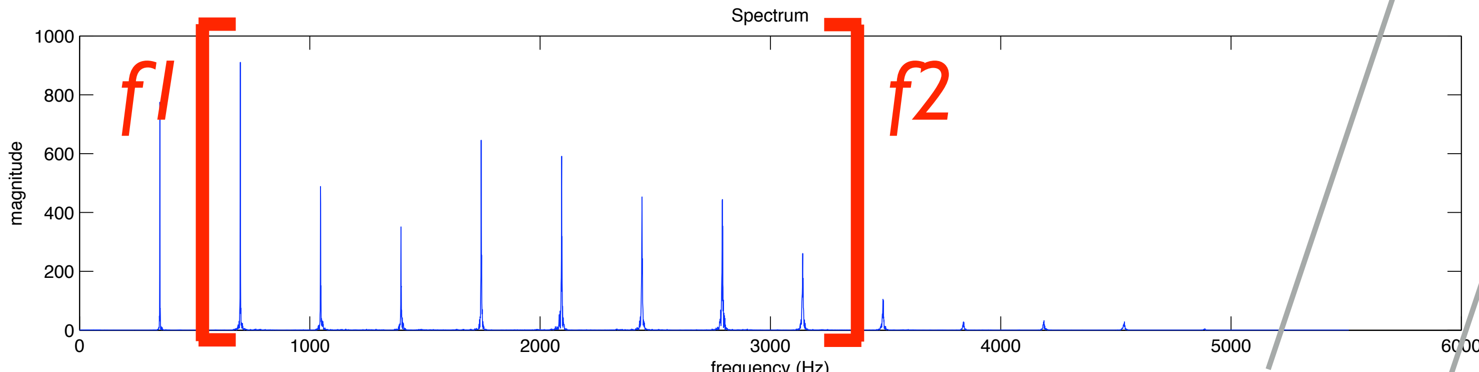
  Silent frames have *RMS* amplitude below *t* times the medium RMS amplitude of the whole audio file.

2. Basic signal processing operators

mirspectrum('trumpet.wav')

Discrete Fourier Transform

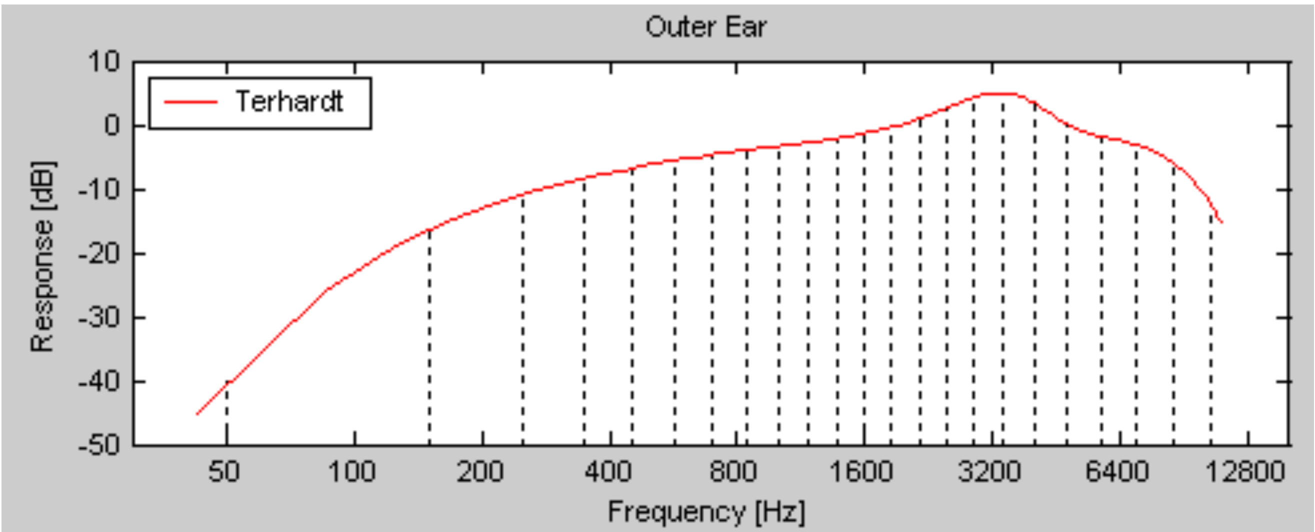
of audio signal *x*: $$X_k = \left| \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} kn} \right|, k = 0, \ldots, N/2$$



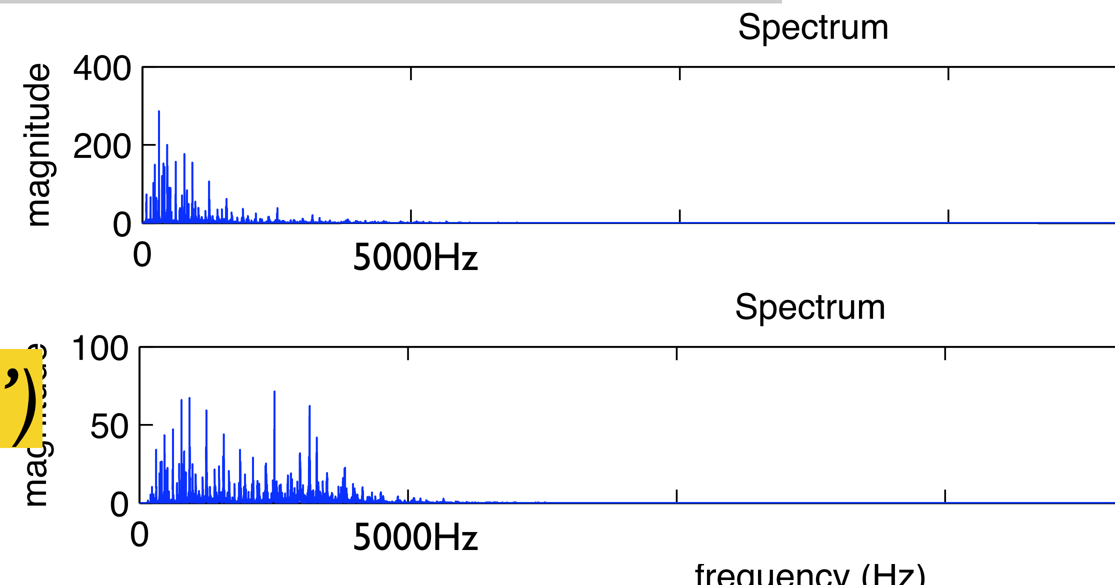
- *mirspectrum(..., '**Min**', f1)* f1=0 Hz
- *mirspectrum(..., '**Max**', f2)* f2=sampling rate/2
- *mirspectrum(..., '**Window**', 'hamming')*

mirspectrum(..., '**Terhardt**')

auditory model: outer-ear filter



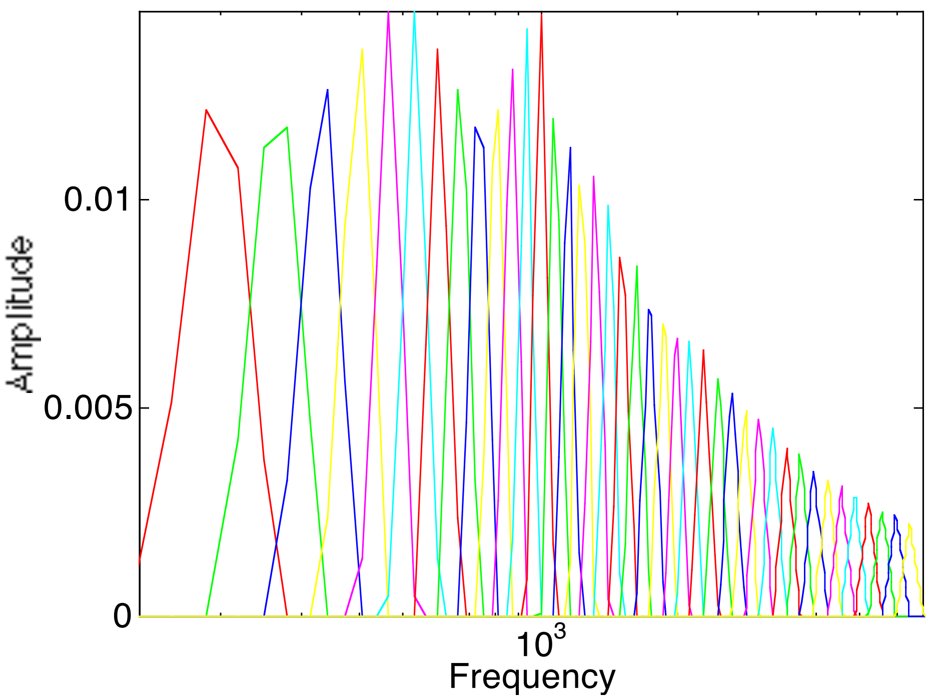
- *mirspectrum*
- *mirspectrum*(..., '***Terhardt***')



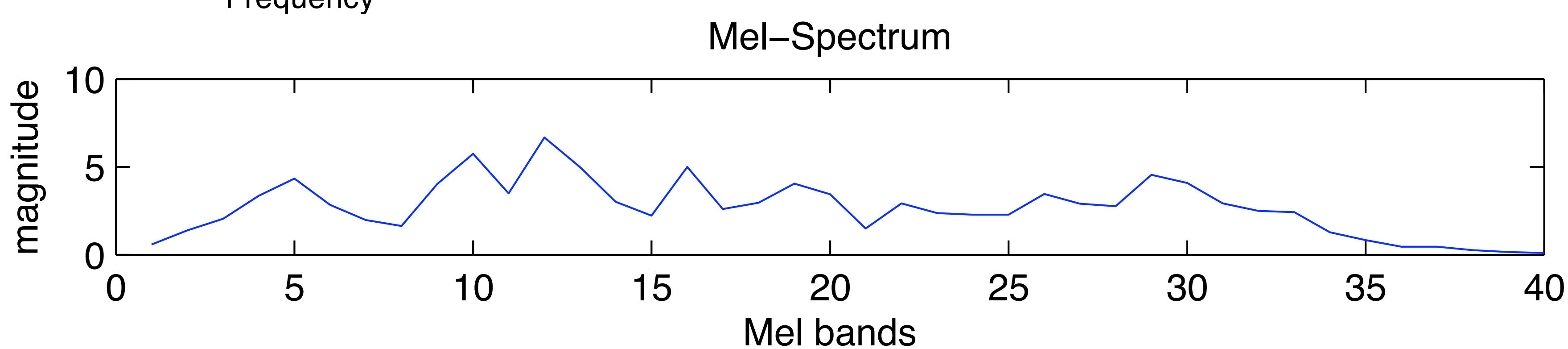
based on *MA* toolbox

mirspectrum(..., 'Mel')

auditory model: Mel-band spectrum



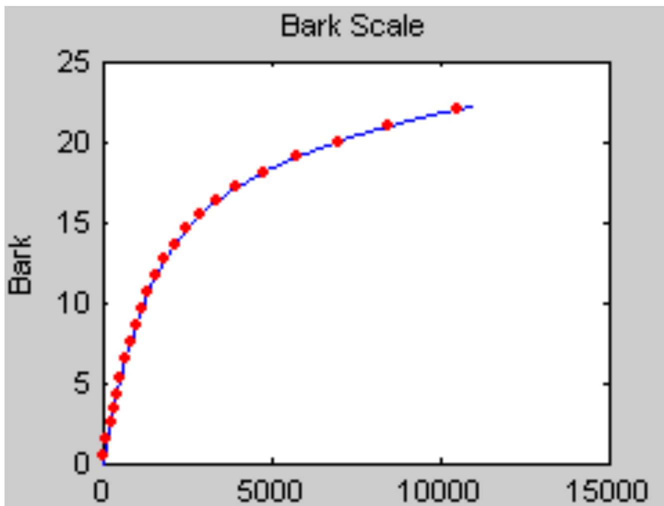
- frequency bands equally spaced on *mel* scale
- in each mel band, perceptually same pitch range



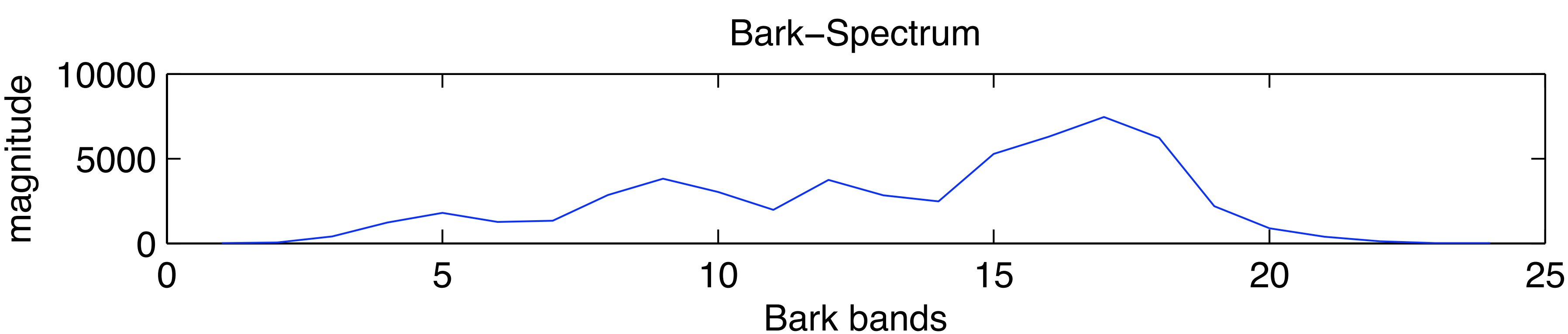
based on *Auditory toolbox*

mirspectrum(..., 'Bark')

auditory model: Bark-band spectrum



- another similar auditory model, decomposing the frequency axis into bands

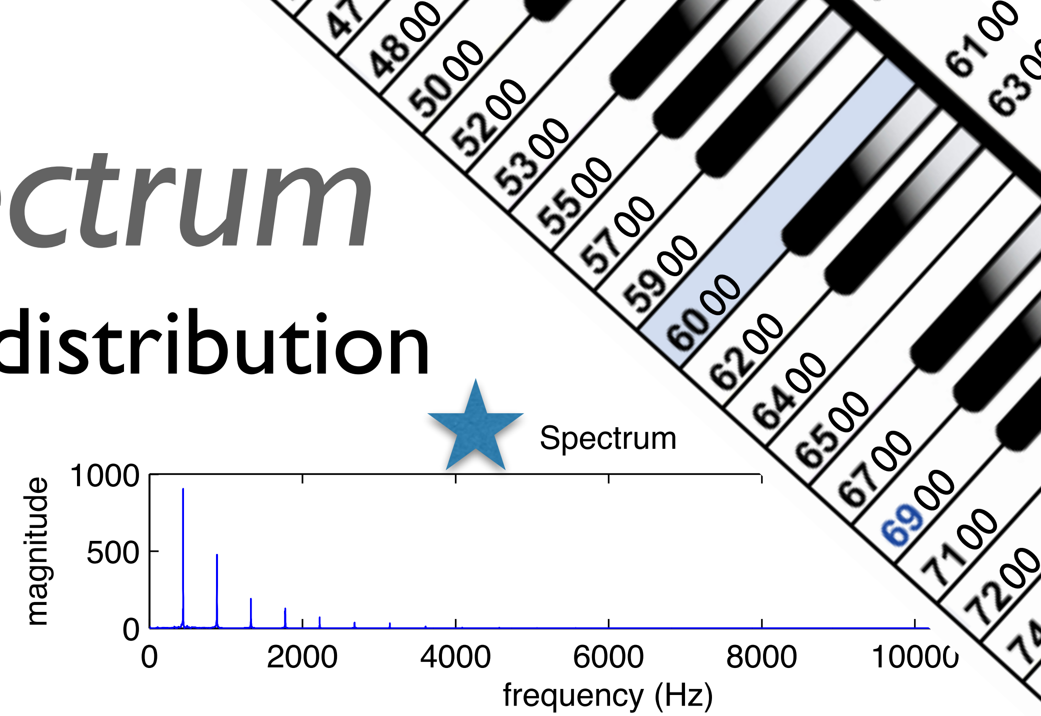


based on *MA* toolbox

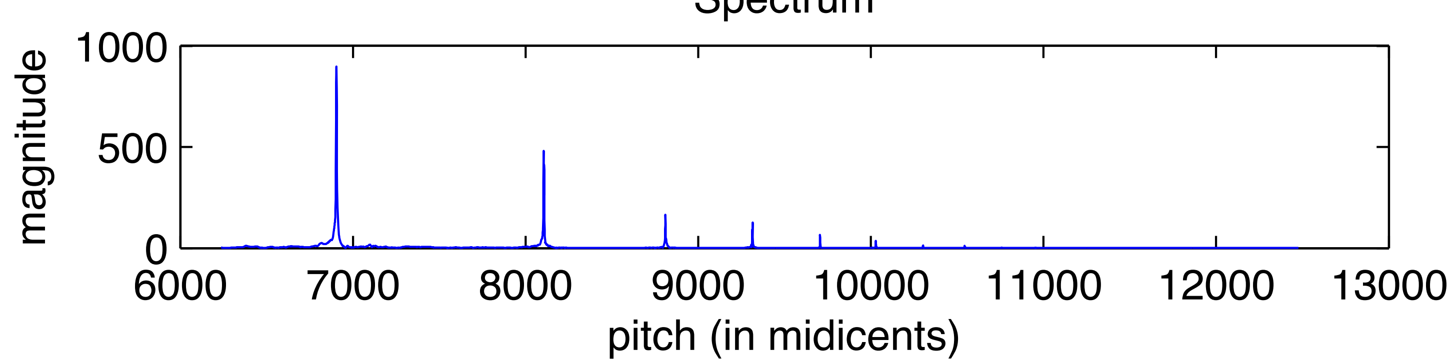
mirspectrum

pitch-based distribution

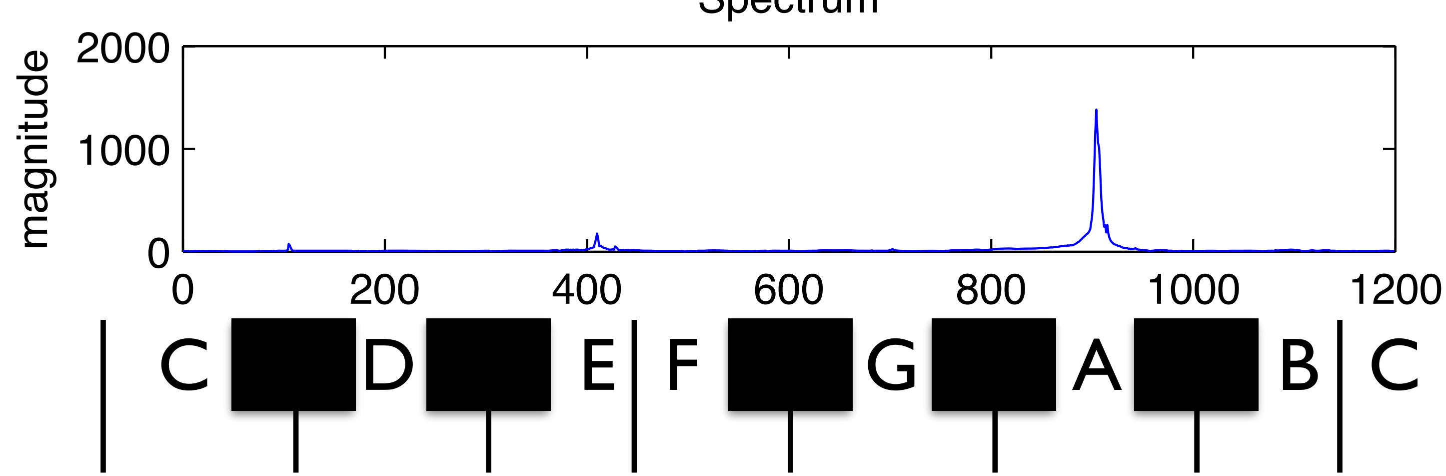
- *mirspectrum('...')*



- *mirspectrum(..., **'Cents'**)*



- *mirspectrum(..., **'Collapsed'**)*

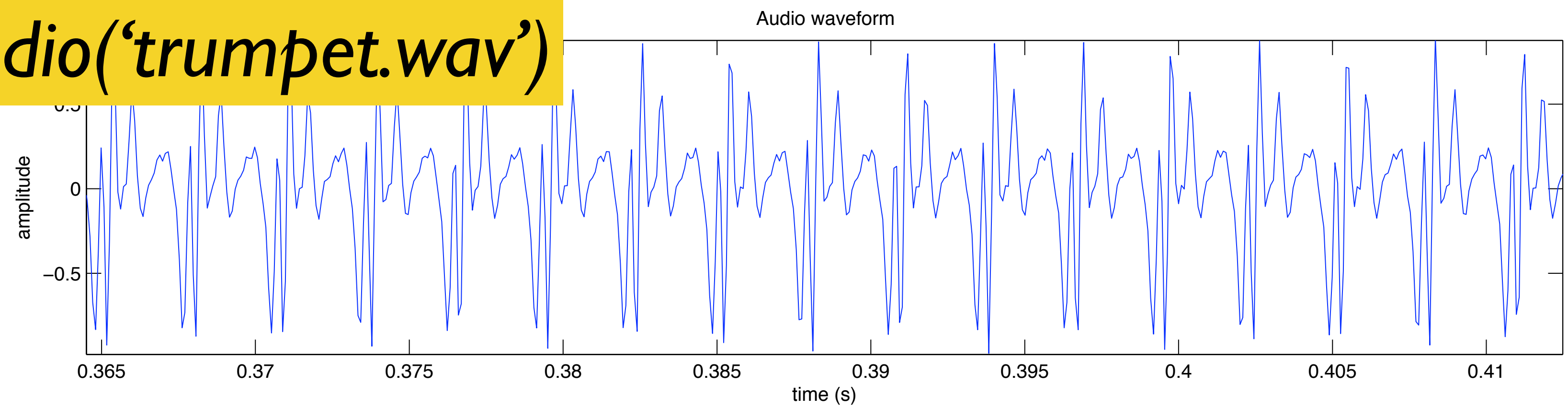


mirautocor

autocorrelation function

miraudio('trumpet.wav')

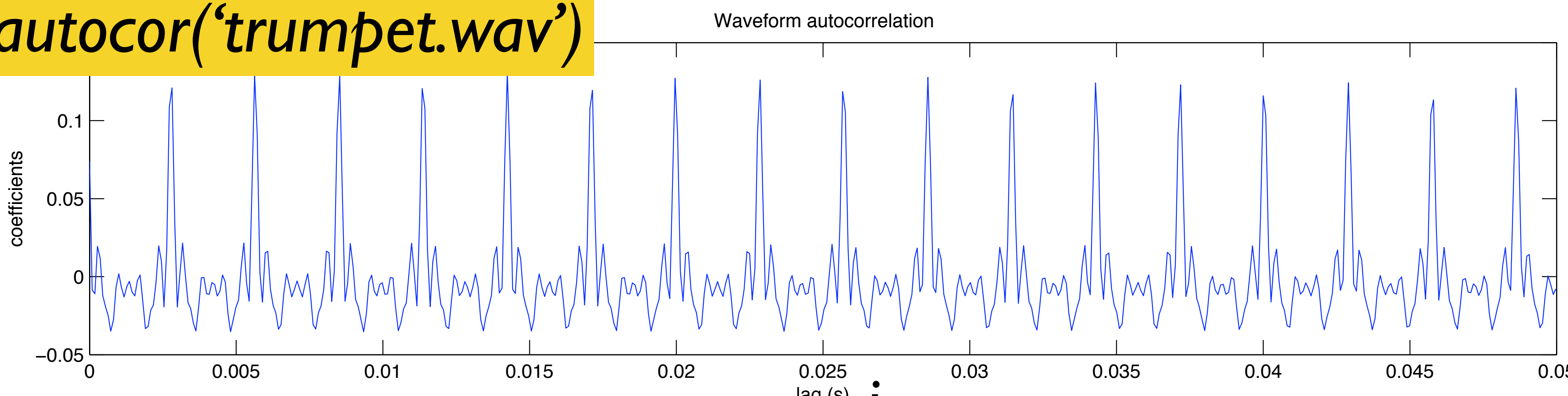
x



$$R_{xx}(j) = \sum_n x_n \overline{x}_{n-j} \ .$$

mirautocor('trumpet.wav')

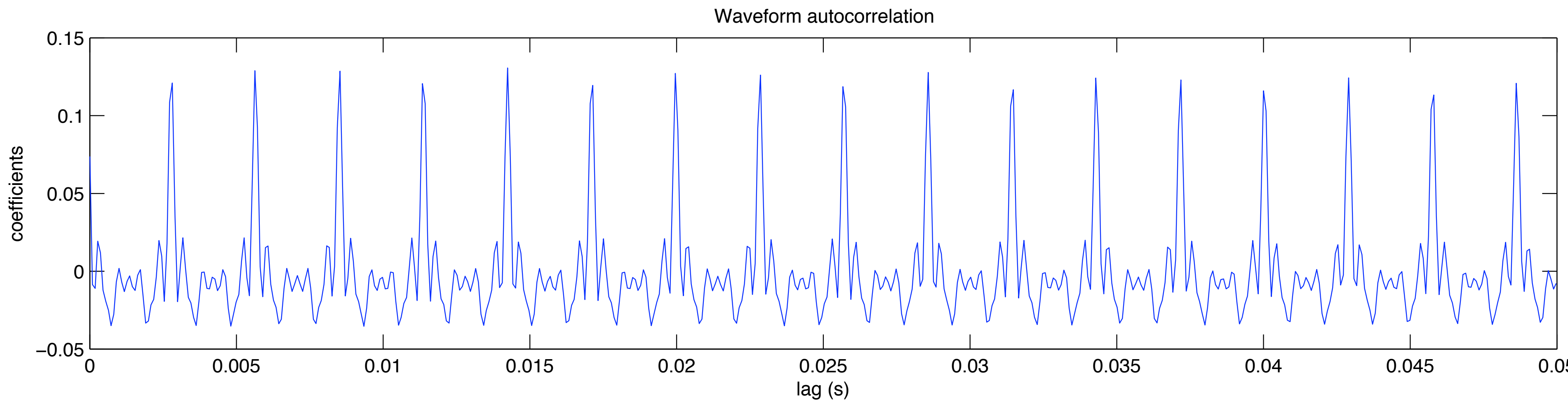
R_{xx}



j

mirautocor

autocorrelation function

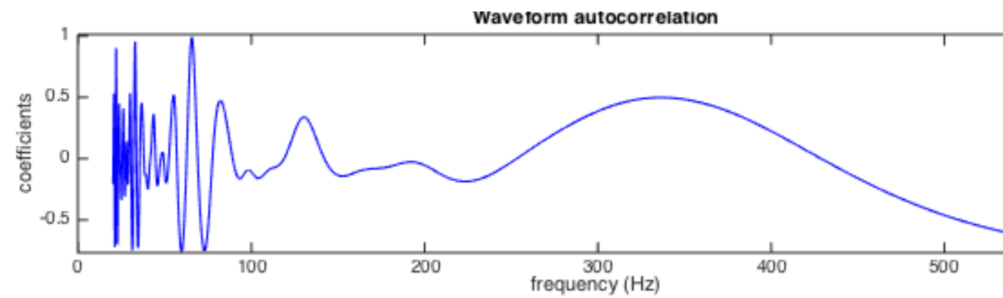


- *mirautocor(…, ‘**Min**’, t1, ‘s’)*   *t1*=0 s
- *mirautocor(..., ‘**Max**’, t2, ‘s’)* *t2*=.05 s (audio) or *t2*=2 s (envelope)
- *mirautocor(..., ‘**Freq**’)*   lags in Hz.

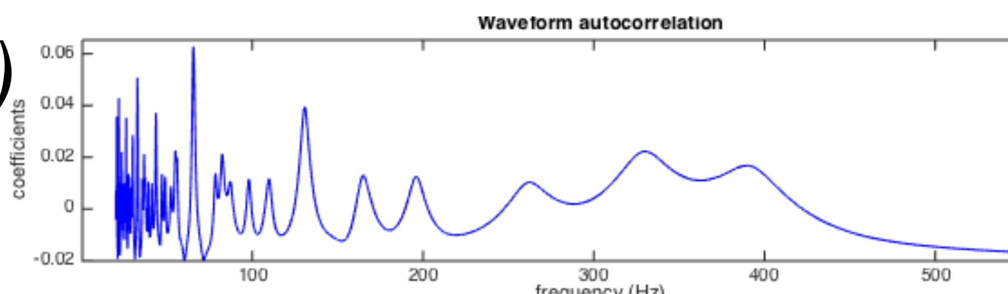
mirautocor(..., 'Compres')

"compressed" autocorrelation

- Autocorrelation (by default): *mirautocor('Cmaj.wav', 'Freq')*
  - $y = IDFT(|DFT(x)|^2)$



- "Compressed" autocorrelation: *mirautocor('Cmaj.wav', 'Freq', 'Compres')*
  - $y = IDFT(|DFT(x)|^{k})$



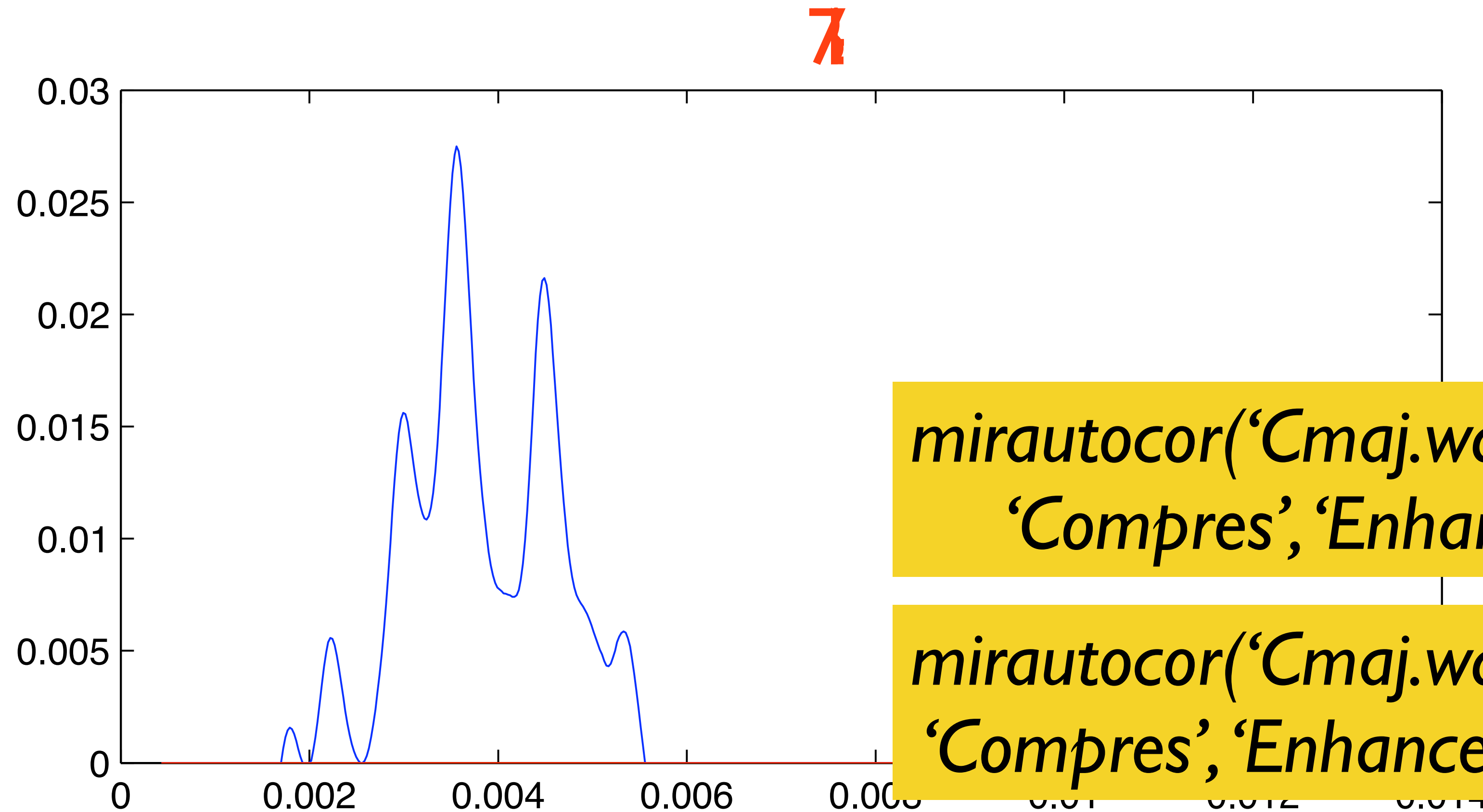
- *mirautocor(…, '**Compres**', k)*
  k=.67

Tolonen, Karjalainen. A Computationally Efficient Multipitch Analysis Model, *IEEE Transactions on Speech and Audio Processing*, 8(6), 2000.

mirautocor(..., 'Enhanced')

enhanced autocorrelation

- *mirautocor('Amin3', 'Enhanced', 2:10)*



mirautocor('Cmaj.wav', 'Freq', 'Compres', 'Enhanced')

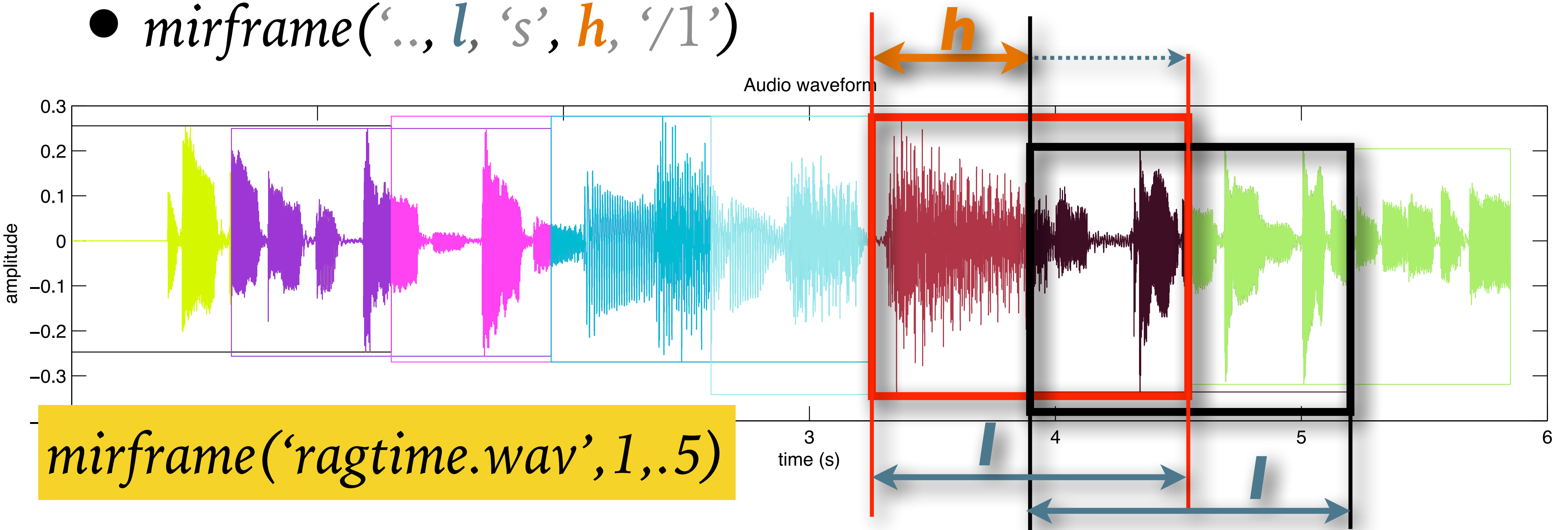
mirautocor('Cmaj.wav', 'Freq', 'Compres', 'Enhanced', 2:20)

Tolonen, Karjalainen. A Computationally Efficient Multipitch Analysis Model, IEEE Transactions on Speech and Audio Processing, 8(6), 2000.

mirframe

frame decomposition

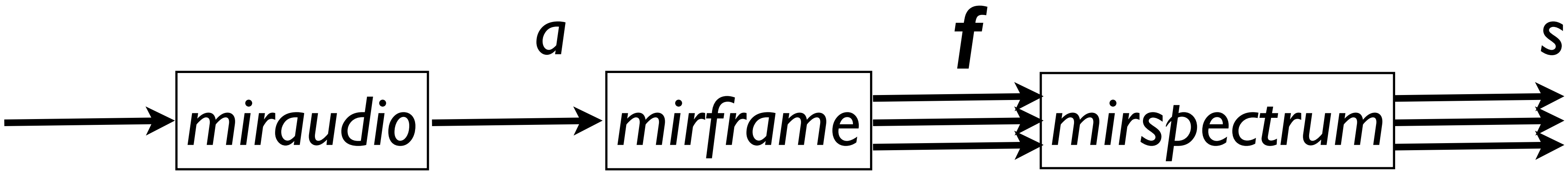
- *mirframe(..., 'WinLength', l, 's')*
  unit: *'s'* (seconds), *'sp'* (samples)
- *mirframe(..., 'Hop', h, '/1')*
  unit: '/1' (ratio from 0 to 1), *'%'* (percentage), *'s'*, *'sp'*
- *mirframe('.., l, 's', h, '/1')*



mirframe('ragtime.wav',1,.5)

mirframe

syntax



a = miraudio('mysong')

f = *mirframe(a)*

f = *mirframe('mysong')*

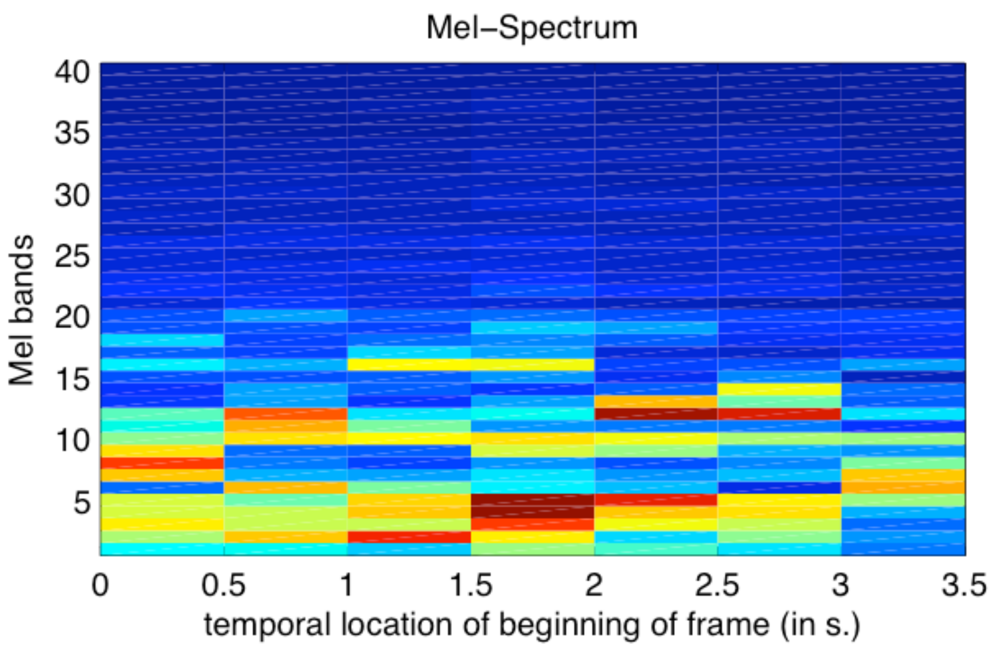
s = mirspectrum(f)

or: *s = mirspectrum('mysong',* ***'Frame'****)*

'Frame' option

syntax

- *miraudio(..., **'Frame'**, l, 's', h, '/1')*
- *mirspectrum(..., **'Frame'**, l, 's', h, '/1')*
- *mirspectrum('mysong', 'Frame', 1, .5, 'Mel')*

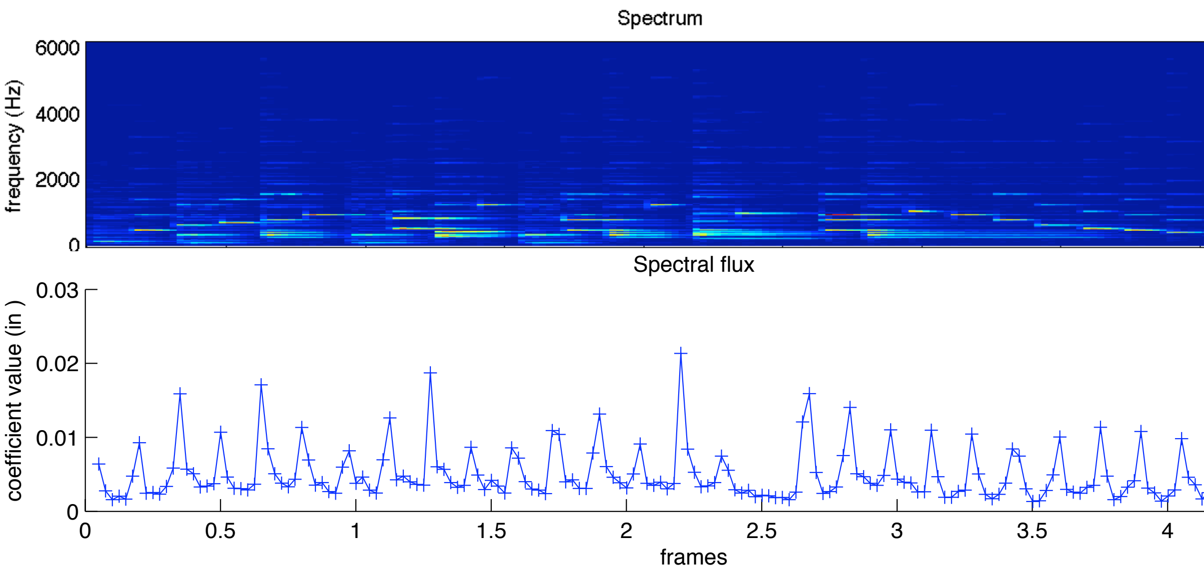


*mir*flux

distance between successive frames

s = mirspectrum(a, 'Frame')

mirflux(s)



- *mirflux(a) = mirflux(mirspectrum(a, 'Frame', .05, .5))*
- *ac = mirautocor(a, 'Frame'), mirflux(ac)*
- *mirflux(..., '**Dist**', d)*   *d = 'Euclidean', 'City', 'Cosine'*

*mir*rms

root mean square

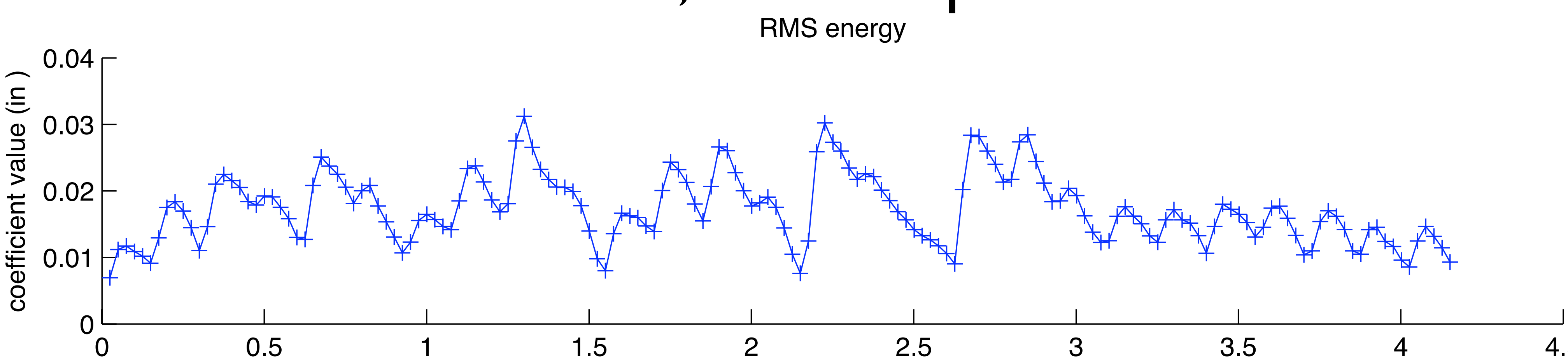
$$x_{\mathrm{rms}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2} = \sqrt{\frac{x_1^2 + x_2^2 + \cdots + x_n^2}{n}}$$

mirrms('ragtime.wav')

```
The RMS energy related to file ragtime is 0.017932
```

mirrms('ragtime.wav', 'Frame')

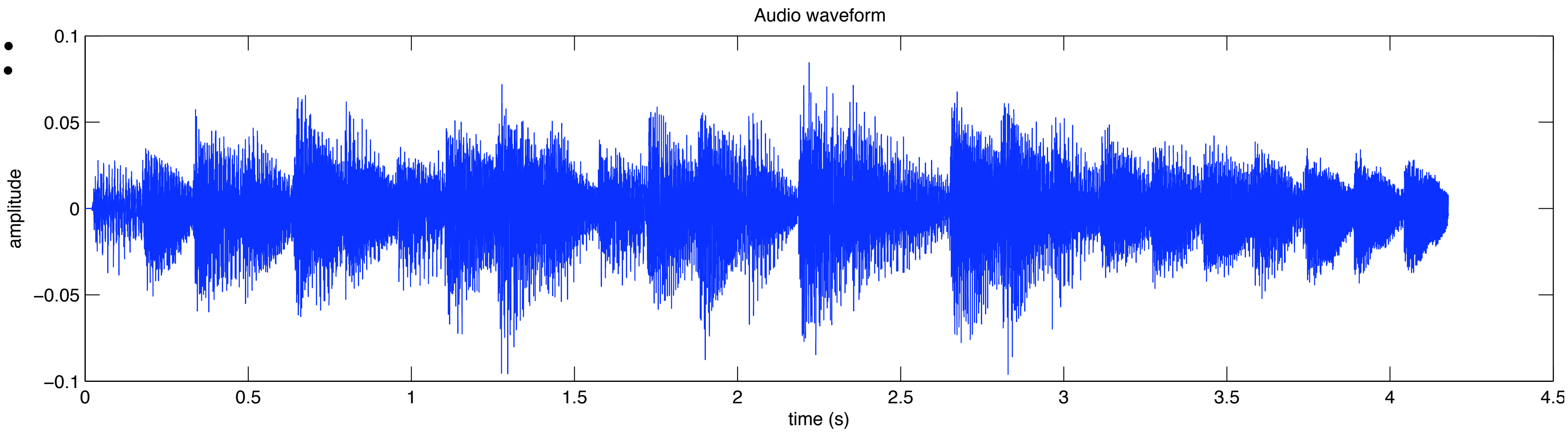
Default frame size .05 s, frame hop = .5



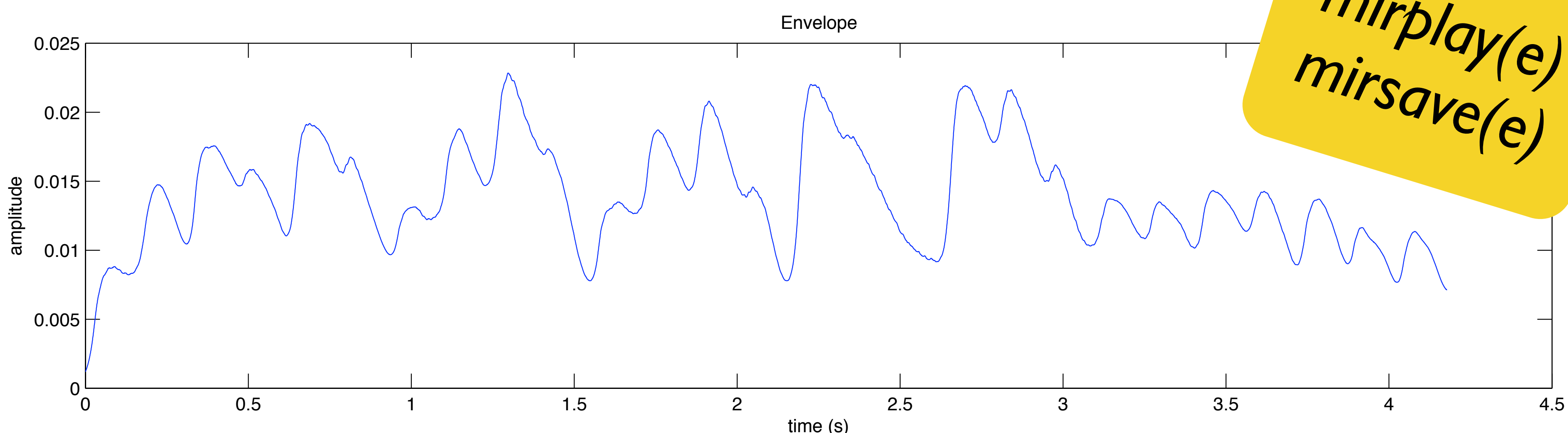
*mir*envelope

envelope extraction

a:

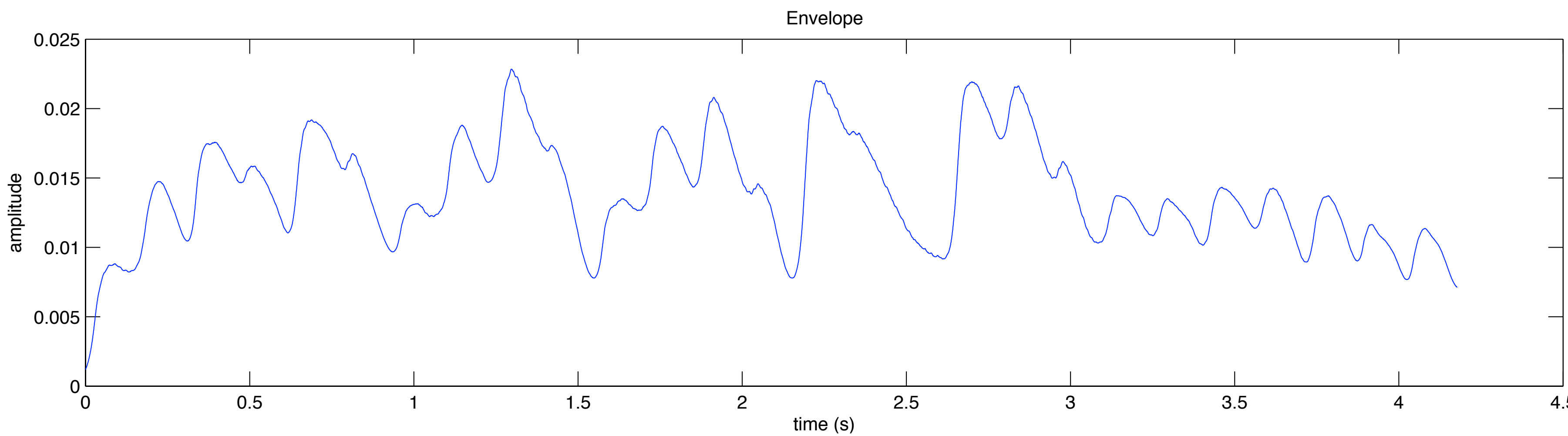


e = *mirenvelope(a)*

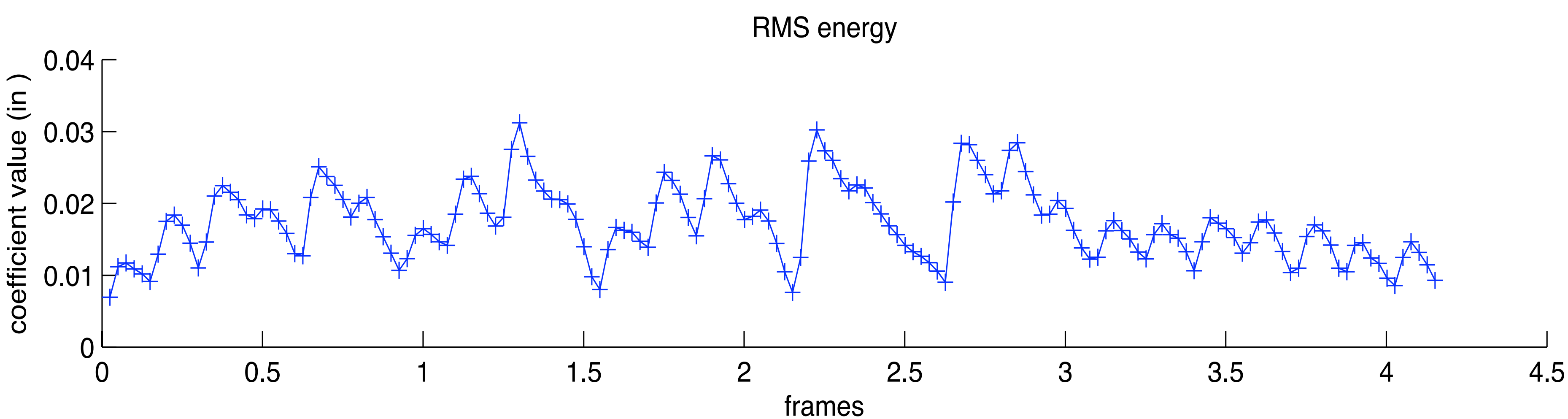


mirplay(e)
mirsave(e)

e = *mirenvelope(a)*

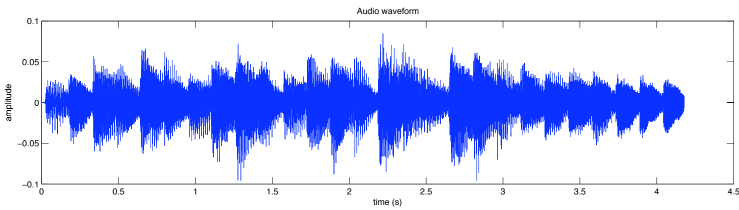


mirrms(a, 'Frame')



mirenvelope(..., '*Filter*')

based on low-pass filtering



abs Full-wave rectification

LPF Low-Pass Filter

mirenvelope(..., '***Tau***', .02): time constant (in s.)

↓N Down-Sampling

mirenvelope(..., '***PostDecim***', *N*) *N*=16

mirenvelope(..., '***Sampling***', *f*)

mirenvelope

post-processing options

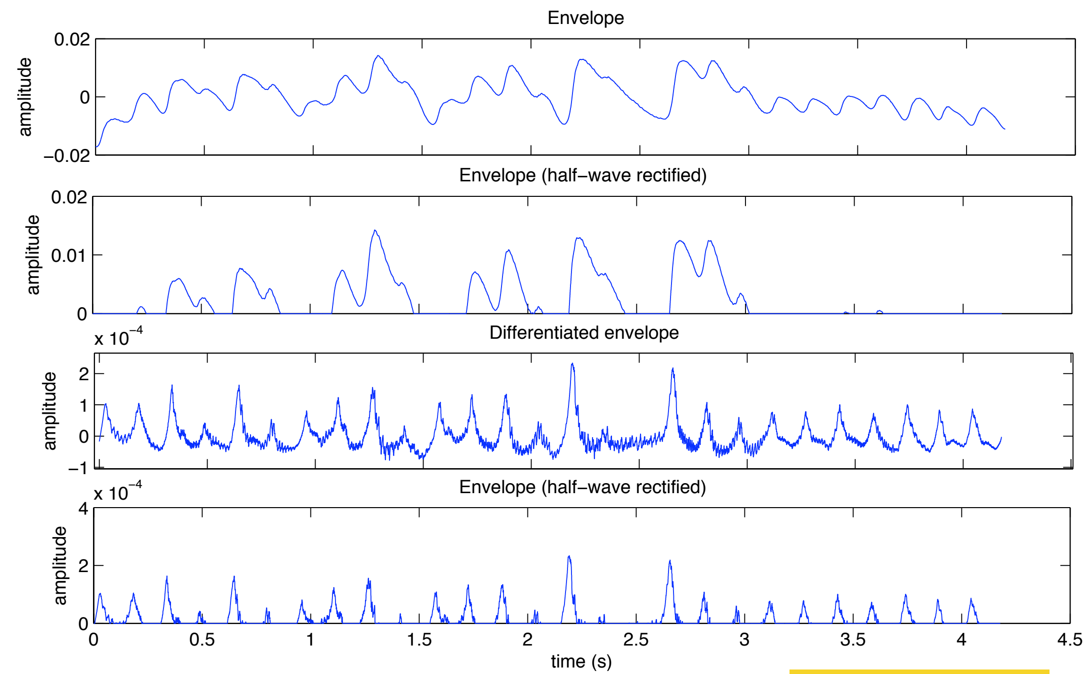
- *mirenvelope(..., '**Center**')*

  '**HalfWaveCenter**')

  '**Diff**')

  '**HalfWaveDiff**')

- *mirenvelope(..., '**Power**')*
- *mirenvelope(..., '**Normal**')*
- *mirenvelope(..., '**Smooth**', o)* moving average, order *o* = 30
- *mirenvelope(..., '**Gauss**', o)* gaussian, std deviation *o* = 30 sp



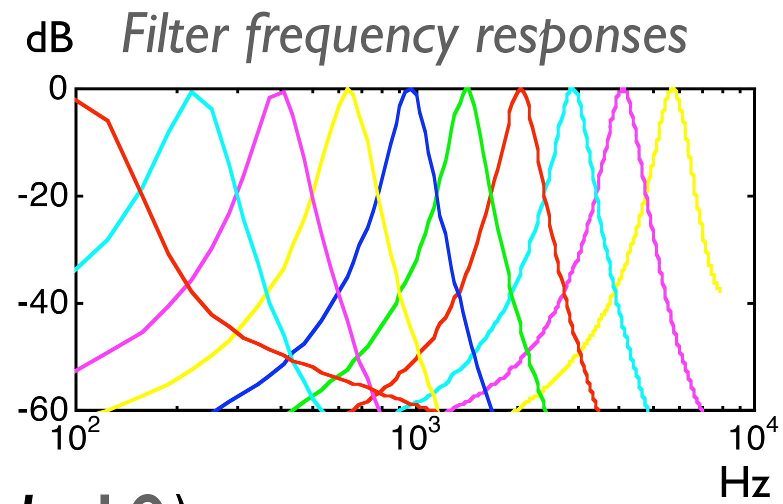
mirplay

mirfilterbank

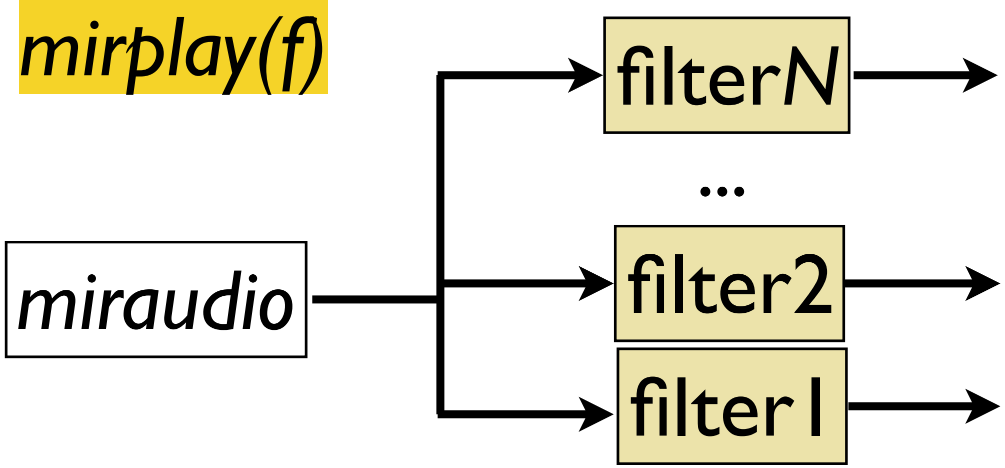
filterbank decomposition

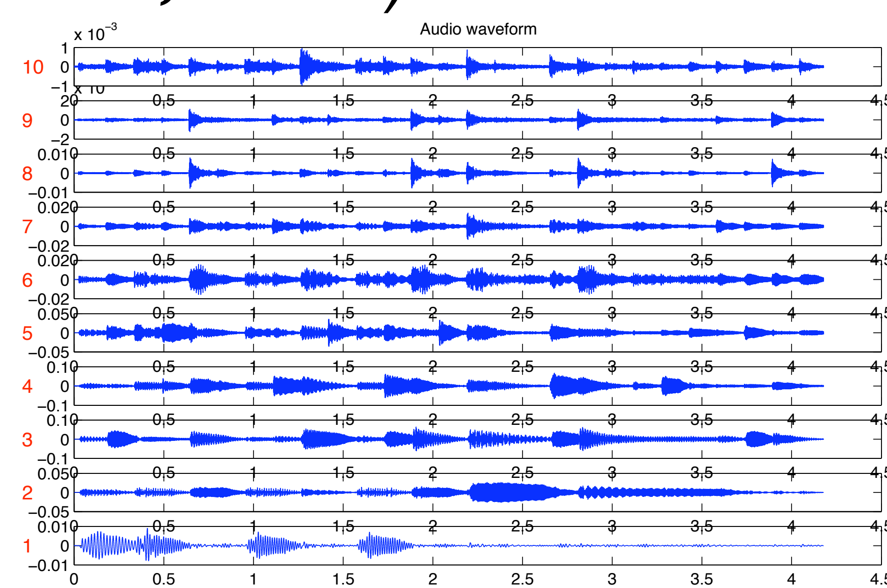
- *mirfilterbank(..., '**Gammatone**')*

  Equivalent Rectangular Bandwidth (ERB) Gammatone filterbank



- f = *mirfilterbank(..., '**NbChannels**', N*=10)
- *mirplay(f)*

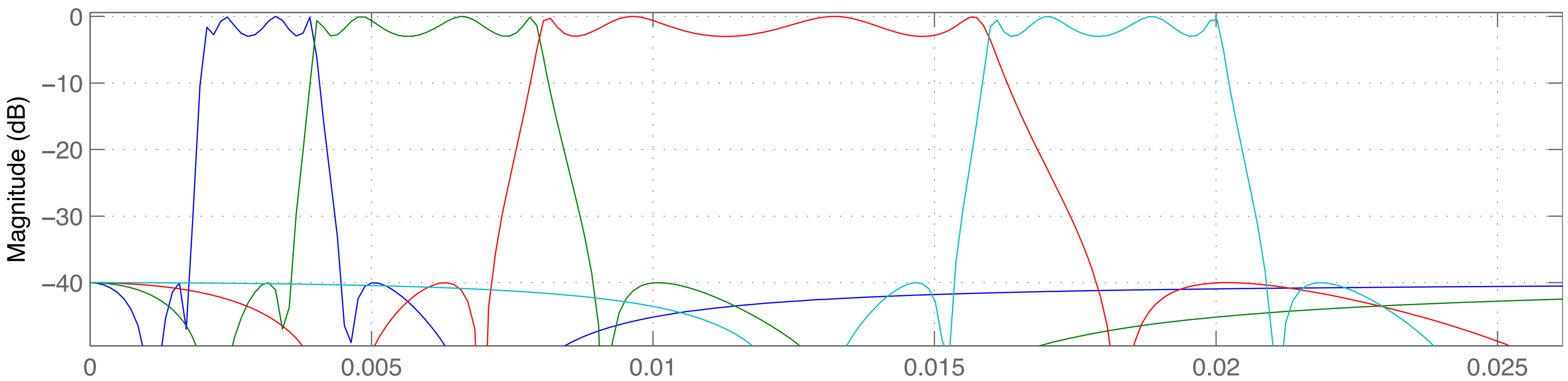




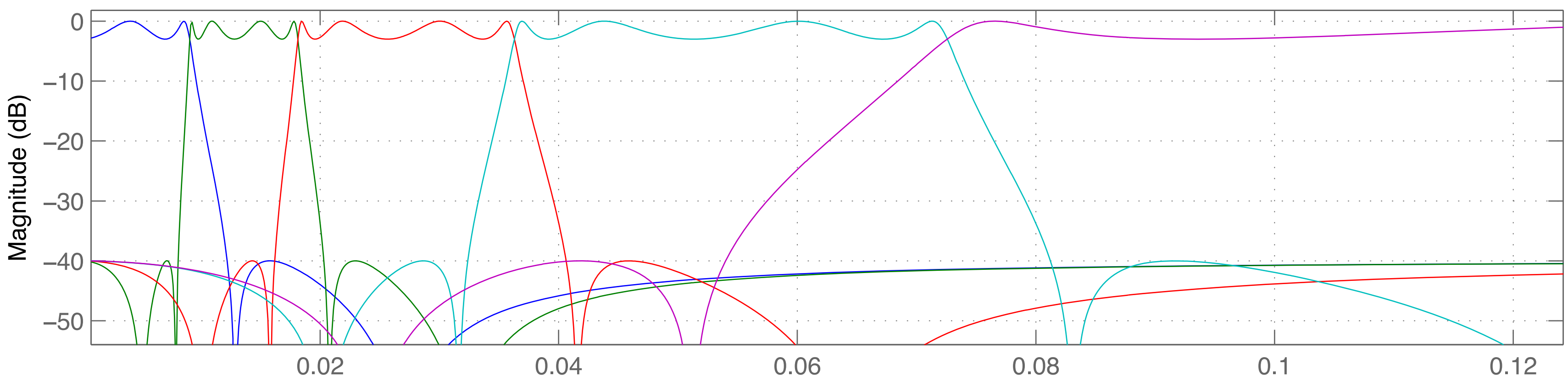
mirfilterbank

filterbank decomposition

- *mirfilterbank(..., '**Manual**', [44, 88, 176, 352, 443])*

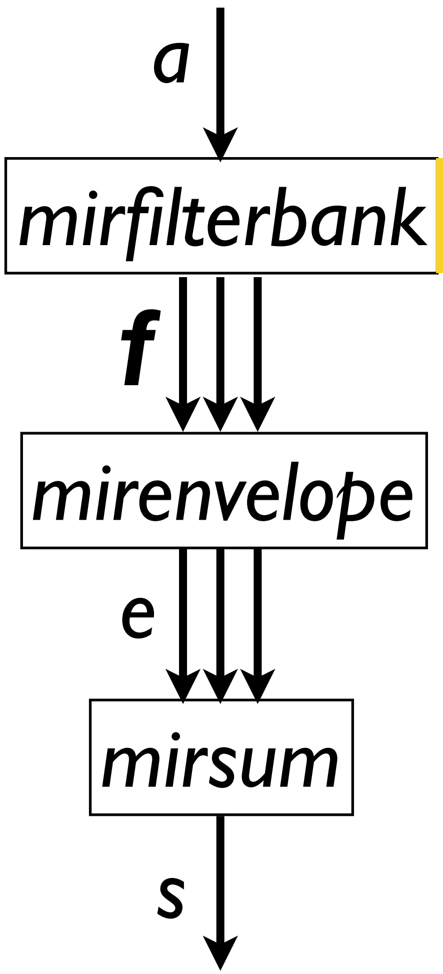


- *mirfilterbank(..., '**Manual**', [-Inf 200 400 800 1600 Inf])*



*mir*sum

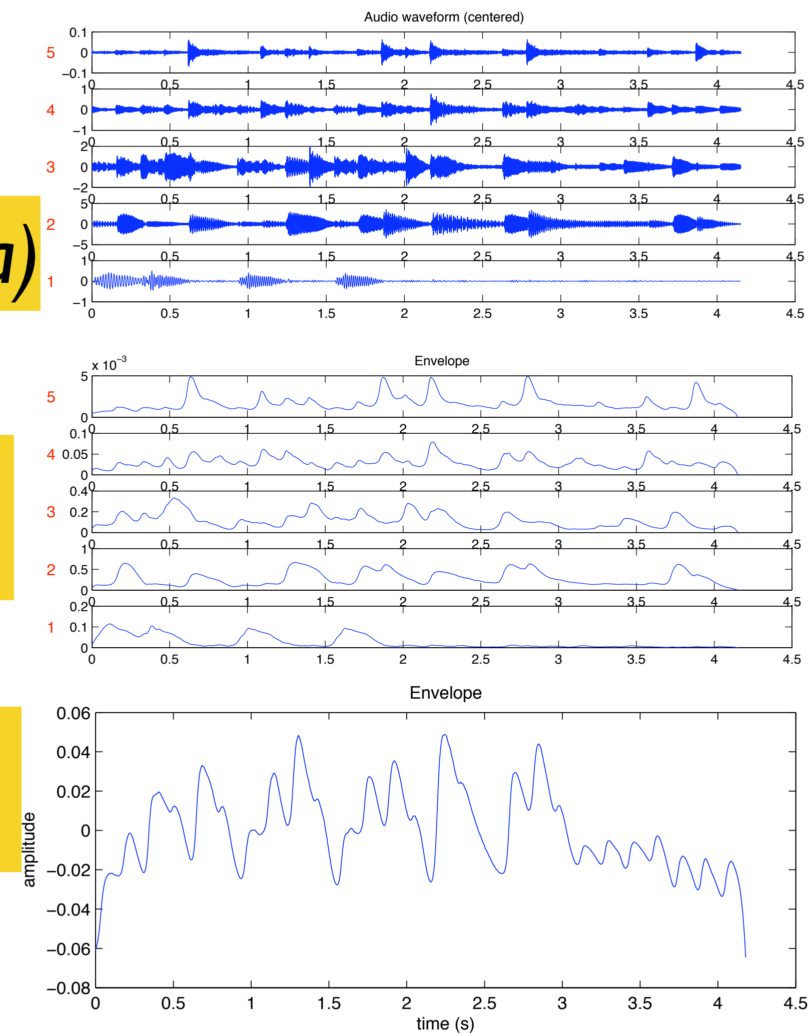
across-channels summa*tion*



f = *mirfilterbank(a)*

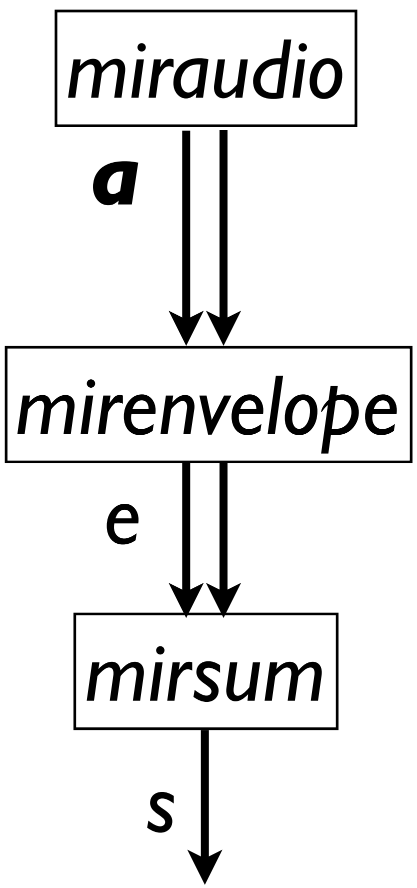
e = *mirenvelope(**f**)*

s = *mirsum(e)*



*mir*sum

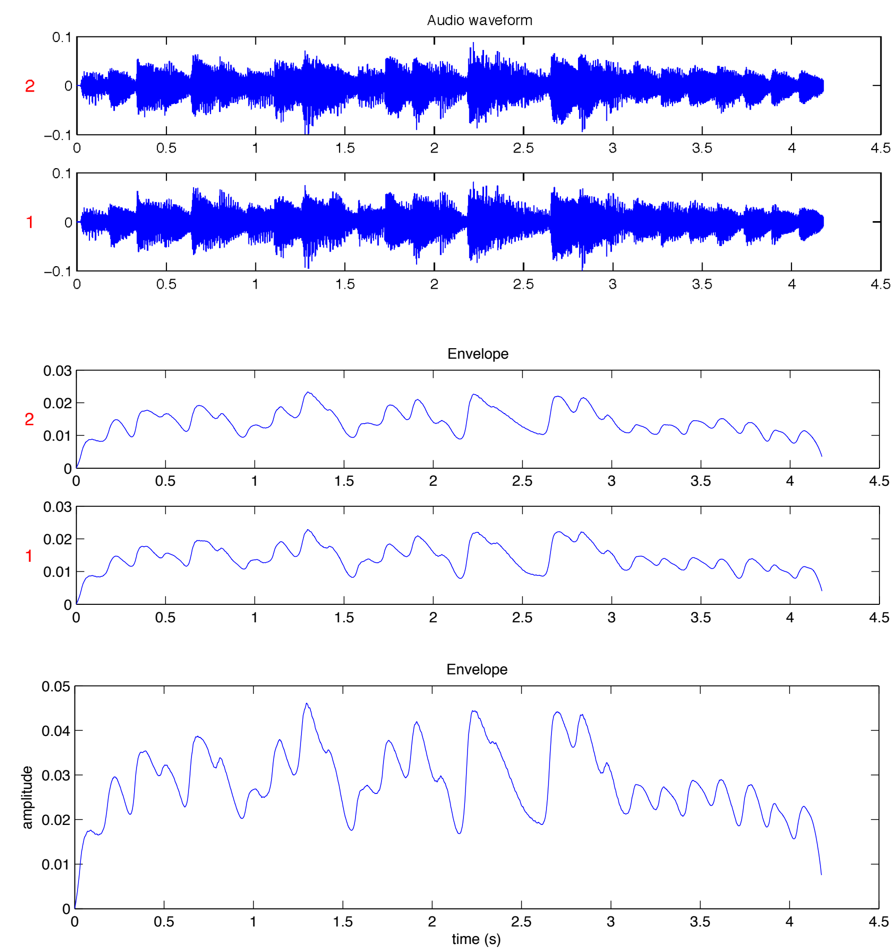
stereo summa*tion*



a = *miraudio(...,*
'Mono', 0)

e = *mirenvelope(**a**)*

s = mirsum(e)



*mir*sum

summary

a

mirfilterbank — ***f*** *= mirfilterbank(a)*

f

mirenvelope — e = *mirenvelope(**f**)*

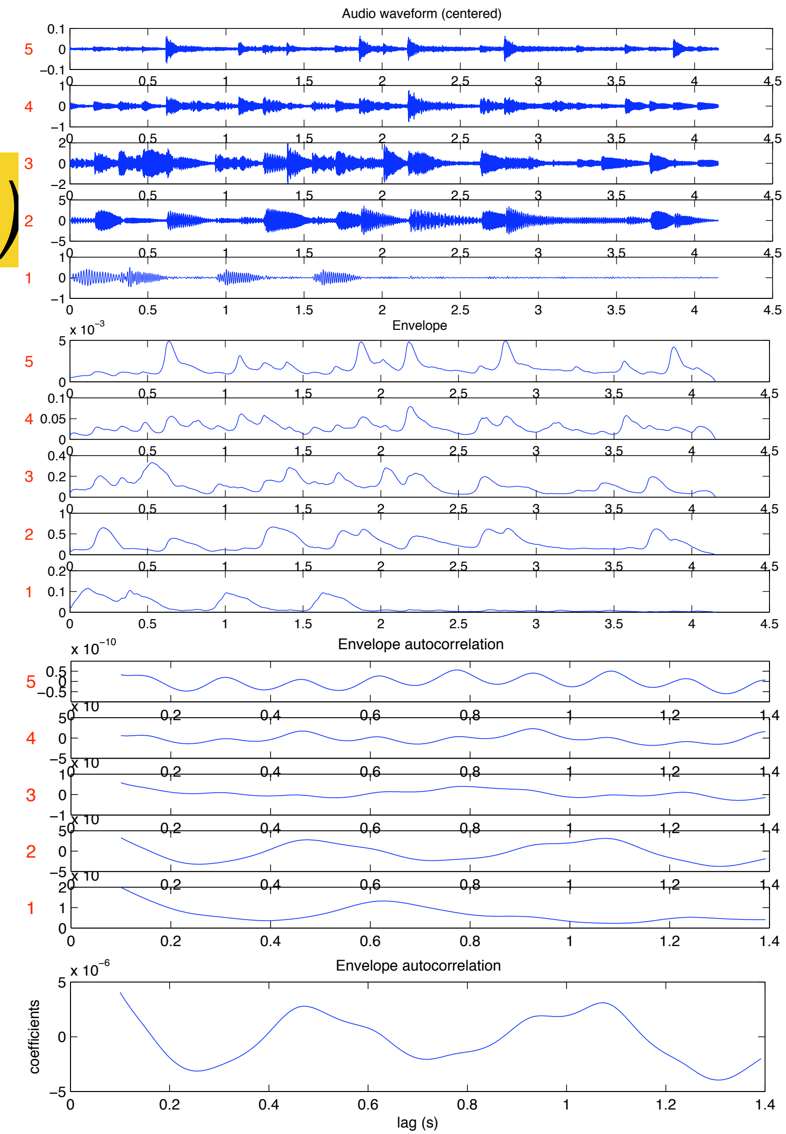
e

mirautocor — *ac = mirautocor(e)*

ac

mirsum — *s = mirsum(e)*

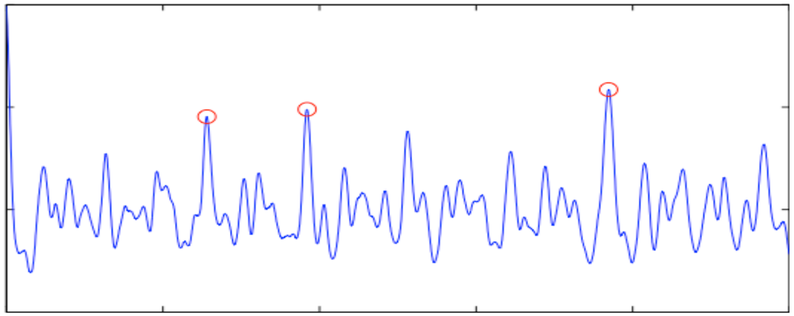
s

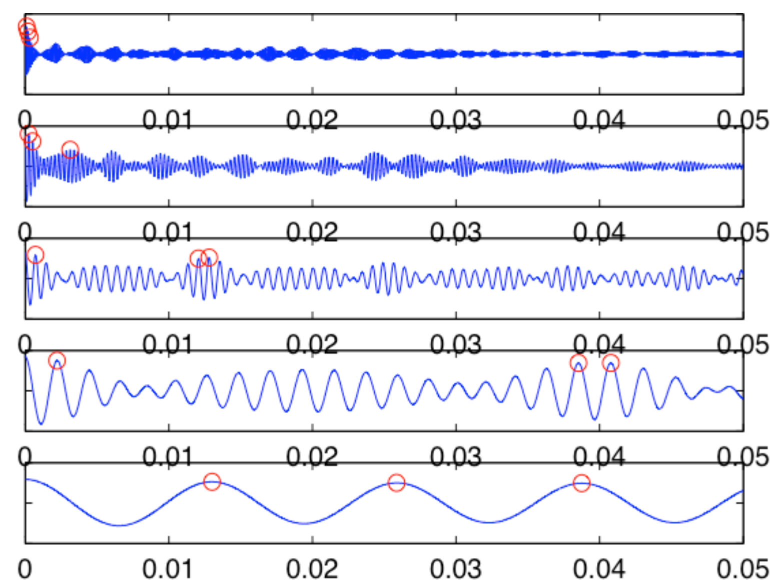


*mir**p**eaks*

peak picking

- *p = mirpeaks(..., 'Total', 3, 'NoBegin')*
- To get peak positions:
  - *mirgetdata(p)*
- To get peak amplitudes:
  - *get(p, 'PeakVal')*





mirpeaks

parameters specification

- *mirpeaks(..., '**Threshold**', t)* default: *t*=0
- *mirpeaks(..., '**Contrast**', c)* default: c=.1

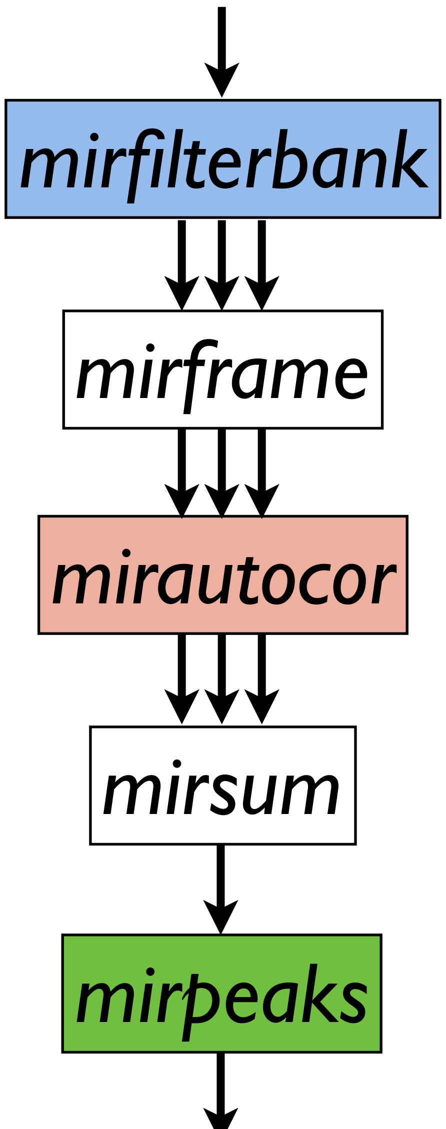
3. Feature extractors

- **Pitch / f0**
- **Timbre**
- Tempo
- Tonality
- Segmentation

(Wednesday)

mirpitch

pitch estimation



mirpitch(...,

'***2Channels***', *or* '***NoFilterbank***',

'***Enhanced***', 2:10,
'***Compress***', .5

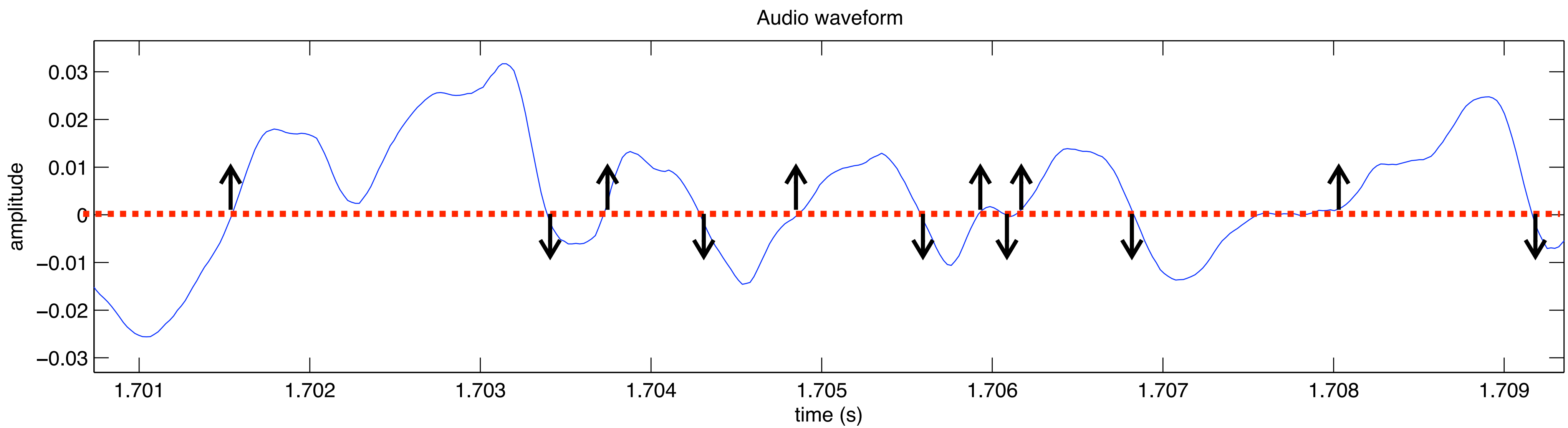
'***Total***', *Inf*,
'***Min***', 75, '***Max***',
2400, '***Contrast***', .1,
'***Threshold***', .4*)*

Timbre

- Zero-crossing rate: *mir**zerocross***
- Spectral distribution: *mir**rolloff***, *mir**brightness***, *mir**centroid***, *mir**spread***, …
- Mel-Frequency Cepstral Coefficients: *mir**mfcc***
- Sensory Dissonance: *mir**roughness***
- *m**irregularity***

*mir*zerocross

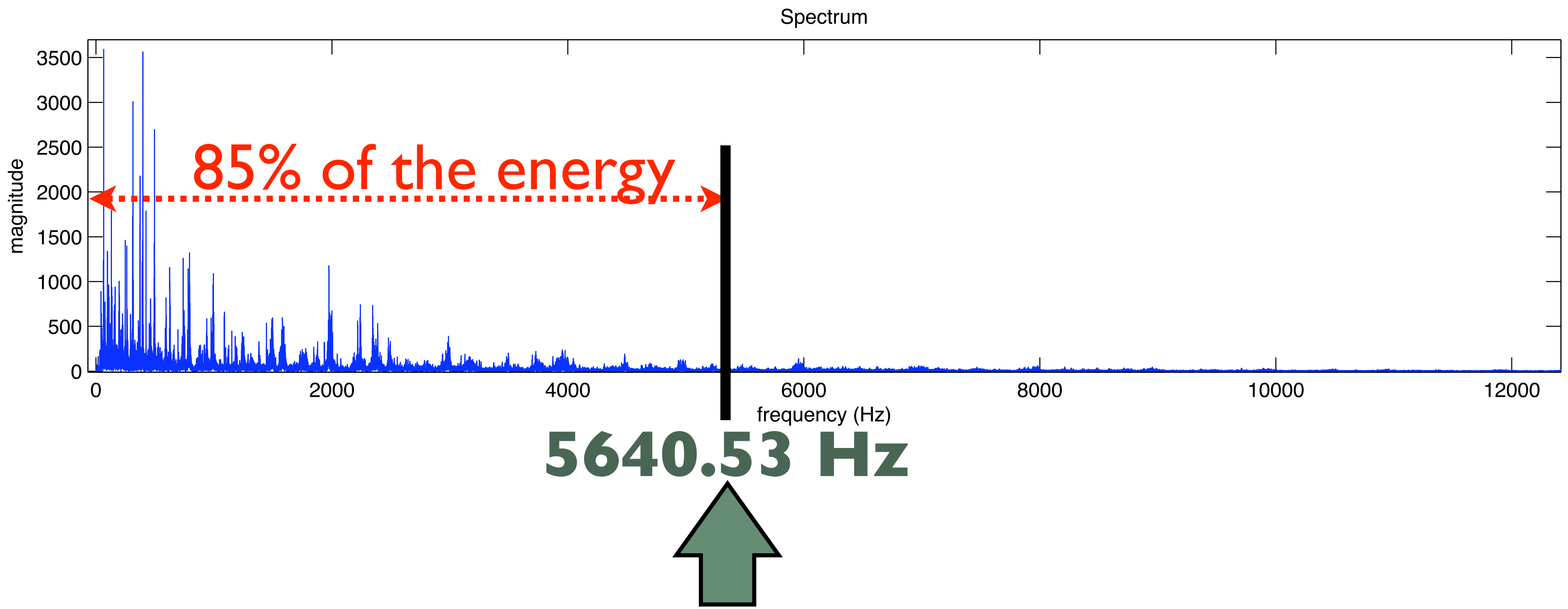
waveform sign-change rate



- Is supposed to indicate how noisy the sound is.
- But highly dependent on the presence of low or high frequency components in the sound.

*mir*rolloff

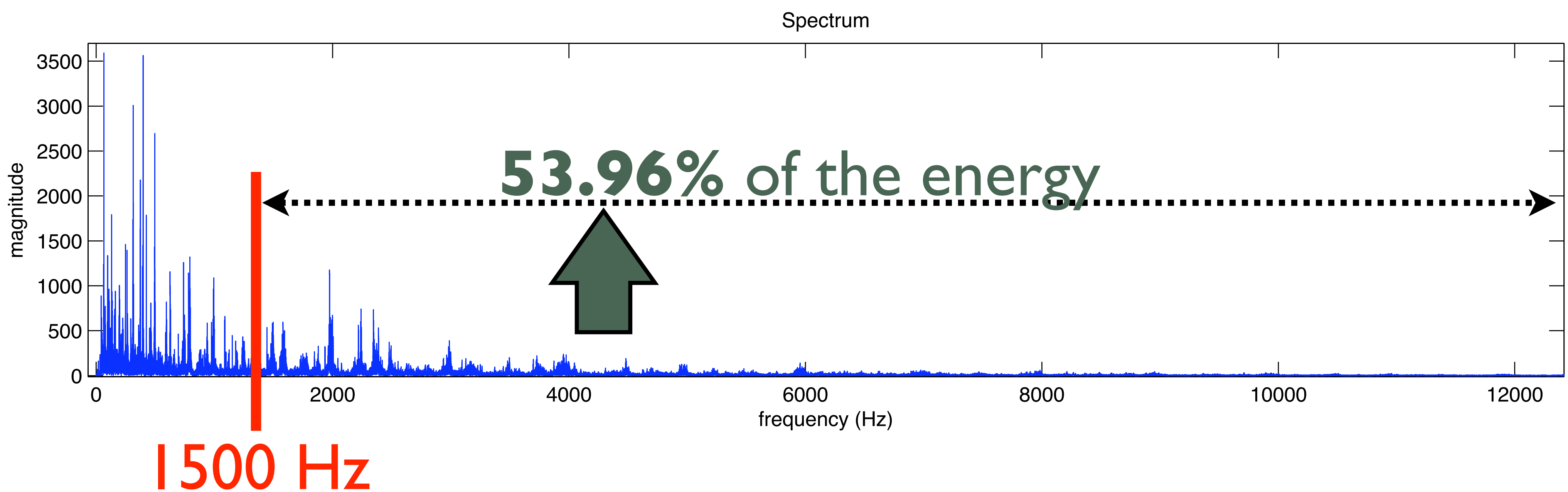
high-frequency energy (I)



- *mirrolloff(..., '**Threshold**', .85)*

*mir*brightness

high-frequency energy (II)

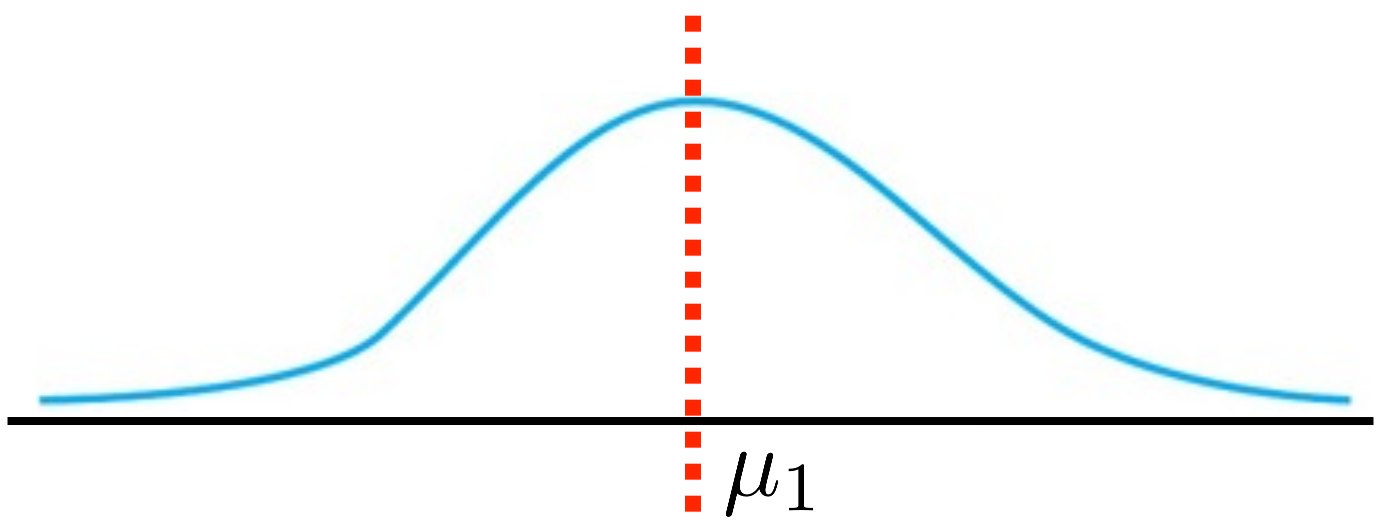


- *mirbrightness(..., '**CutOff**'*, 1500) (in Hz)
- *mirbrightness(..., '**Unit**', u)*   *u* = '/1' *or* '%'

*mir*centroid

geometric center of spectral distribution

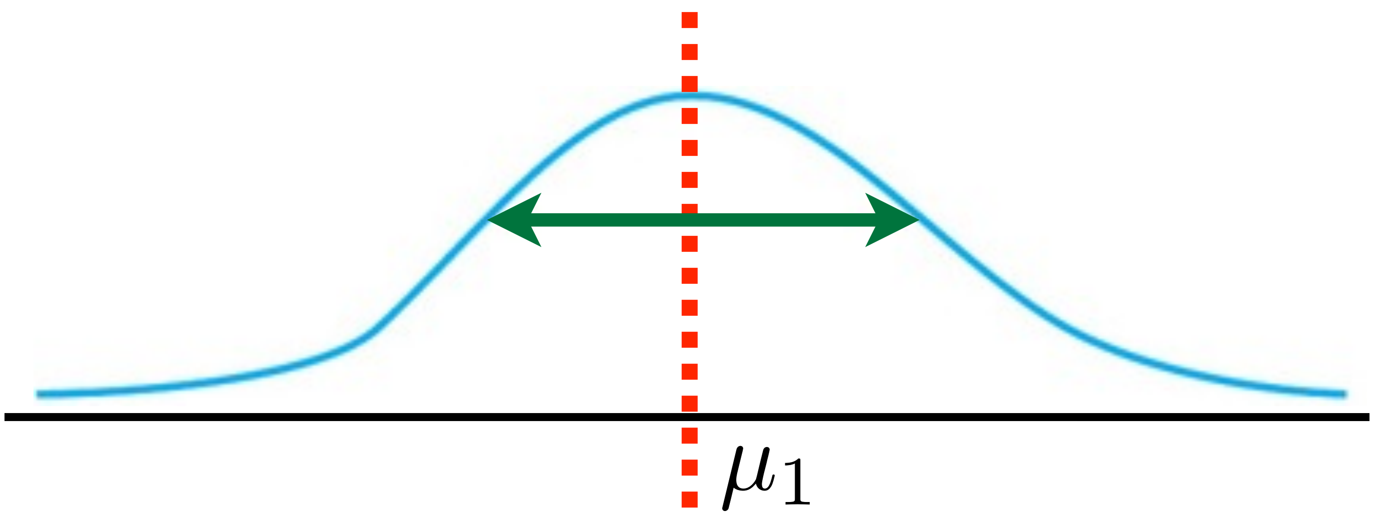
$$\mu_1 = \int x f(x) dx$$



mirspread

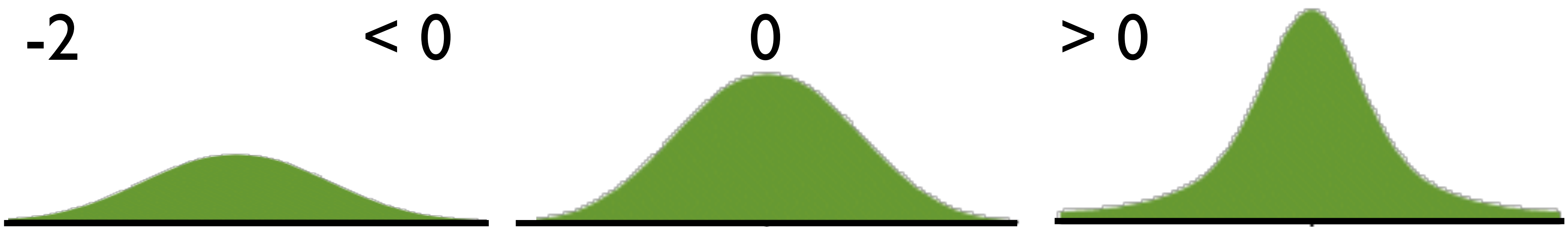
spectral dispersion

second moment: $\sigma^2 = \mu_2 = \int (x - \mu_1)^2 f(x) dx$



*mir*kurtosis

spectral pickiness



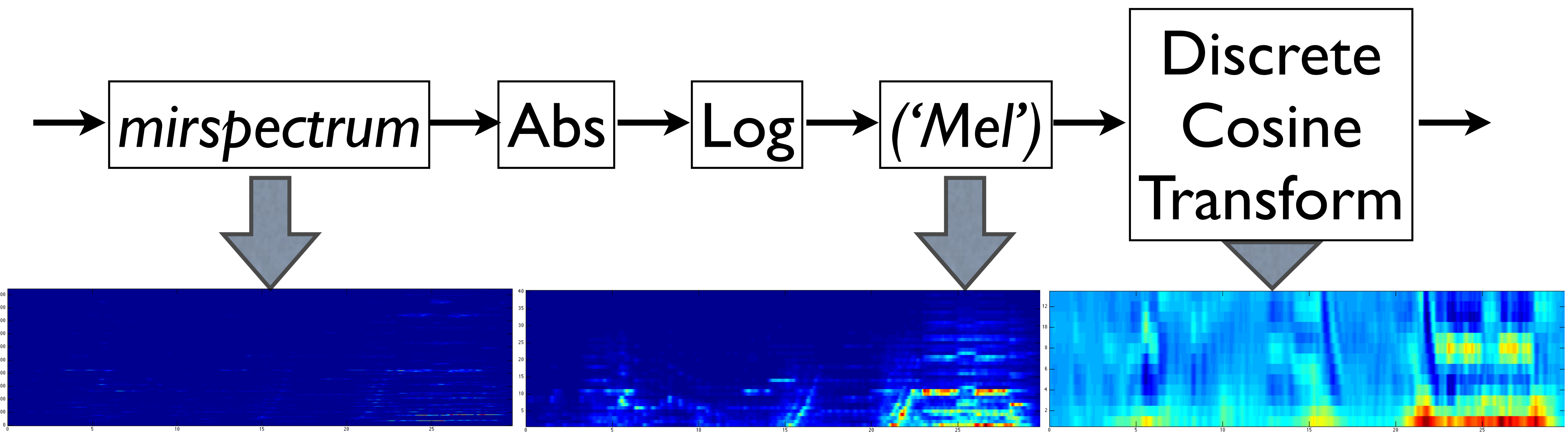
*mir*flatness

smooth vs. spiky

$$\frac{\text{geometric mean}}{\text{arithmetic mean}} \qquad \frac{\sqrt[N]{\prod_{n=0}^{N-1} x(n)}}{\left(\frac{\sum_{n=0}^{N-1} x(n)}{N}\right)}$$

mirmfcc

mel-frequency cepstral coefficients

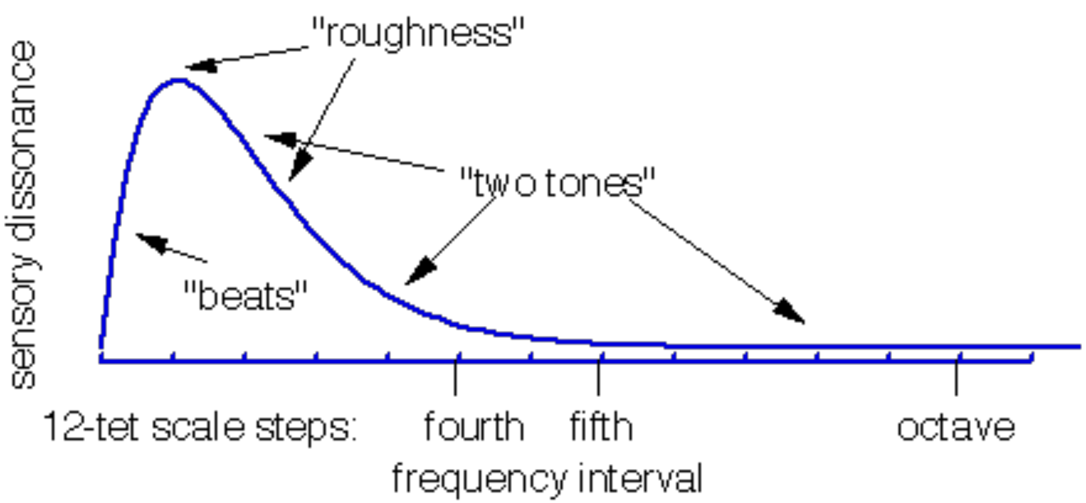


- Description of spectral shape.

*mir*roughness

sensory dissonance

- *mirroughness(..., **'Sethares'**)*

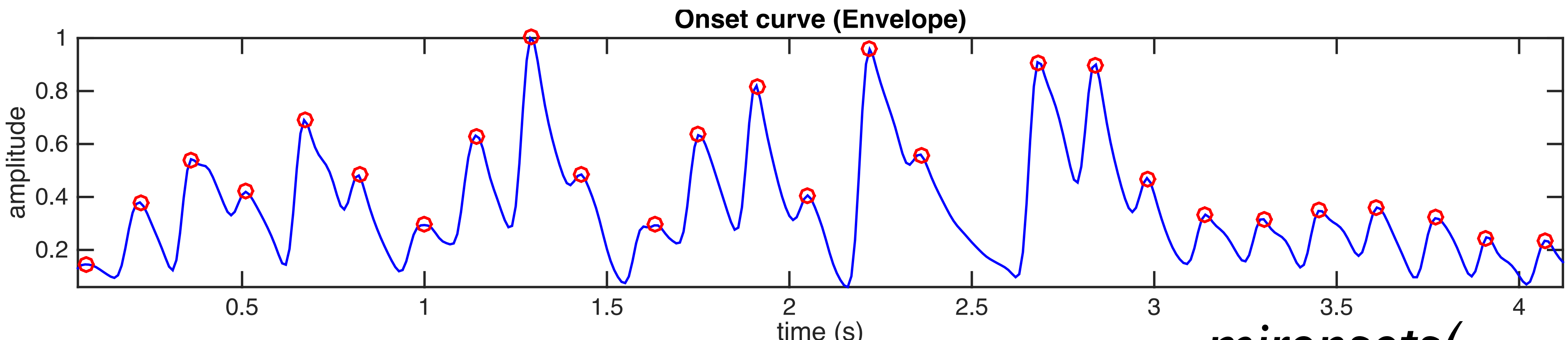


Dissonance produced by two sinusoids depending on their frequency ratio

→ *mirspectrum* → *mirpeaks* → **dissonance** estimated for each pair of peaks → sum →

*mir*onsets

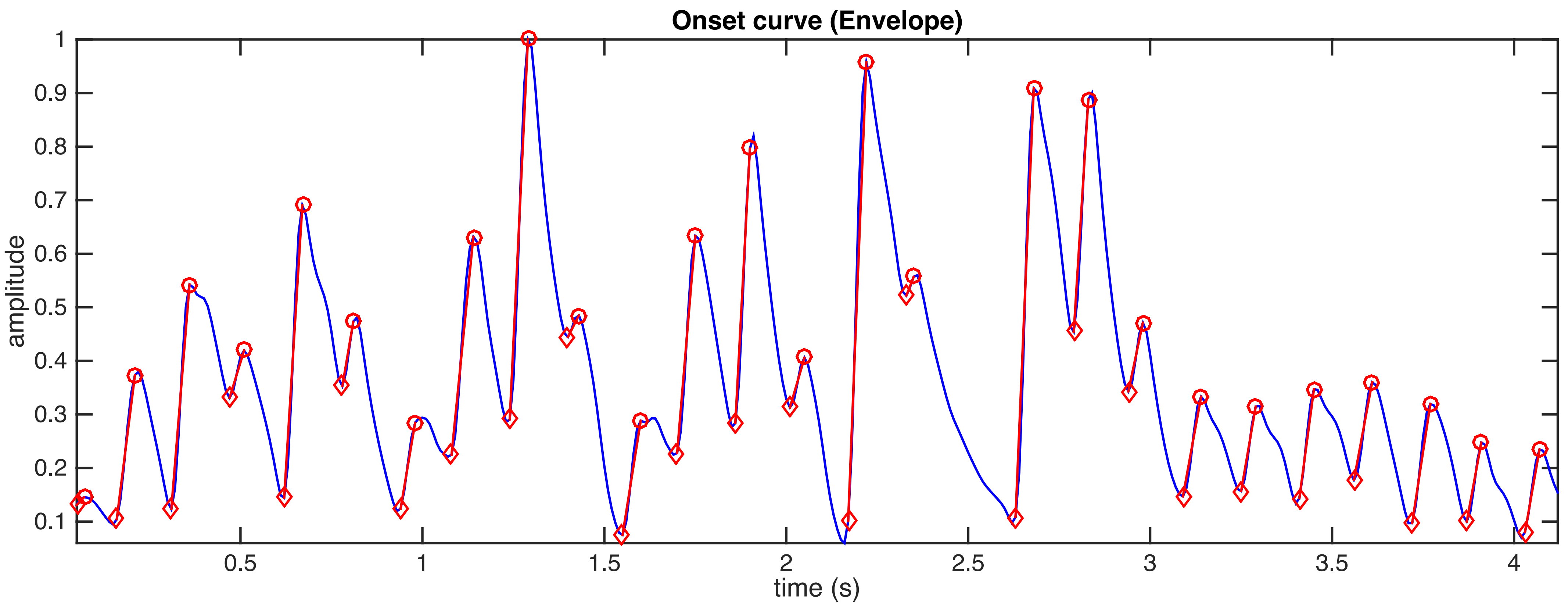
onset detection function



mironsets(…,
'Contrast'*,*
…)

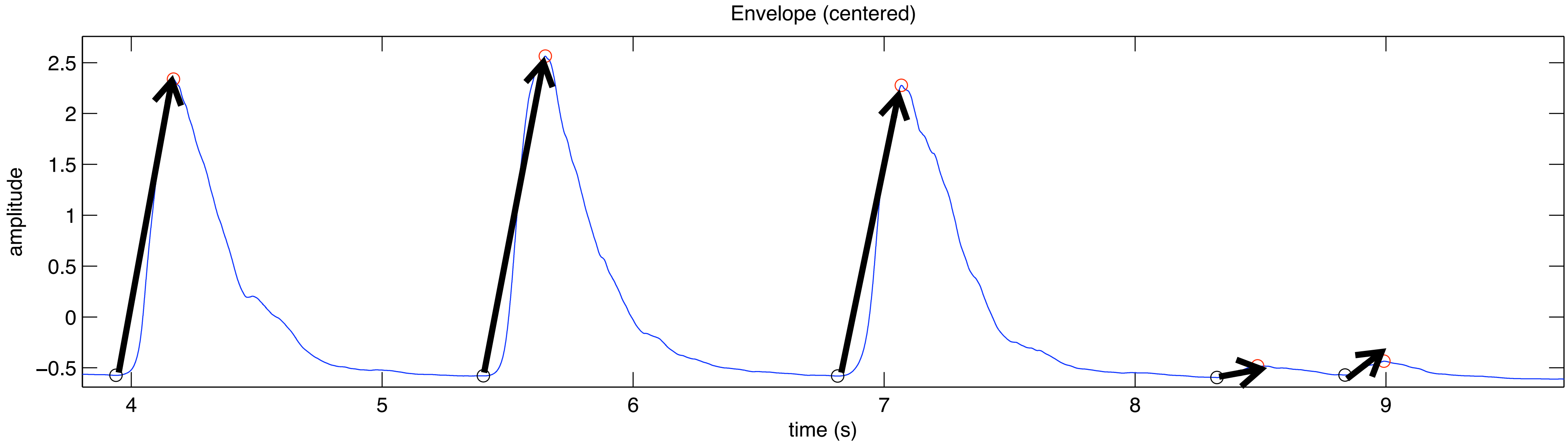
- *mironsets*
  - *mirpeaks(mirsum(mirspectrum(…,'Frame')))*
- *mironsets(…,'**Filter**')*
  - *mirpeaks(mirsum(mirenvelope(mirfilterbank(…,'NbChannels', 40))))*
- *mironsets(…,'**SpectralFlux**')*
  - *mirpeaks(mirflux(…,'Inc','Halfwave'))*

mironsets(..., 'Attack')



mirattackslope

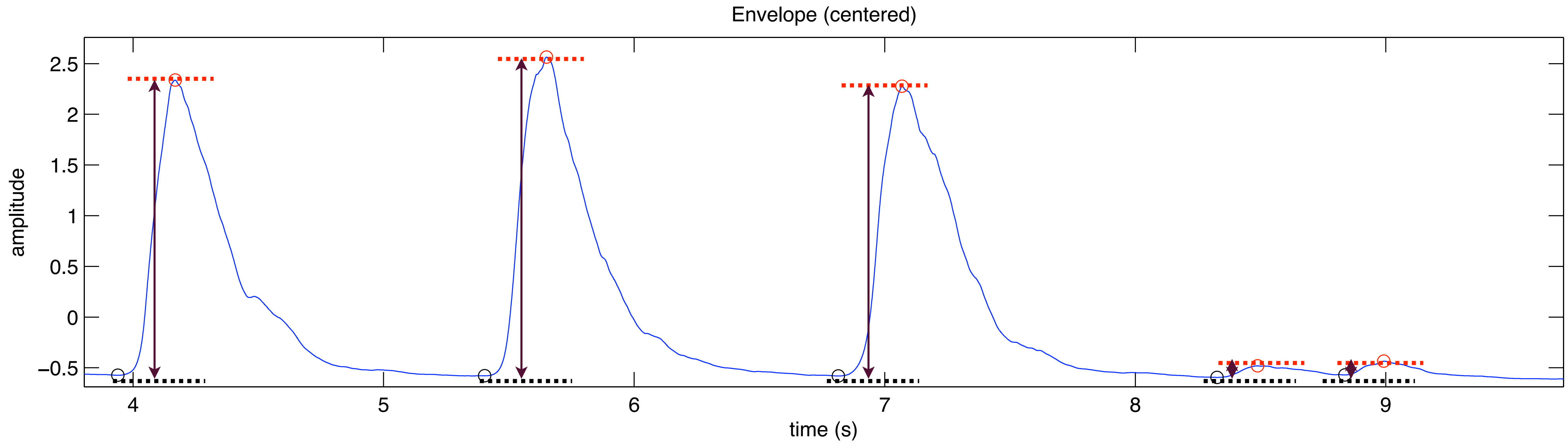
average slope of note attacks



- *o = mironsets('george.wav', ...)*
- *mirattackslope(o)*

mirattackleap

amplitude of note attacks



- *o = mironsets('george.wav', ...)*
- *mirattackleap(o)*

Part 2 (in 2 weeks)

- Rhythm, metrical structure
- Tonal analysis
- Segmentation, structure
- Statistical descriptions, similarity
- Music & emotion
- Advanced use