Perception & Multimedia Computing

Week 19 – Cool effects & Perceptual Audio Features

Michael Zbyszyński
Lecturer, Department of Computing
Goldsmiths University of London

Last time...

Finishing Filters: FEEDBACK!

Multiplication & Convolution

Revisiting FFT

Goals for you

- Reason about different types of LTI systems
  - *Using impulse response*
  - *Using frequency response*
- Use filter design tools to create new filters
- Apply filters & effects in your own code
  - *How?*

Today

Some last audio effects

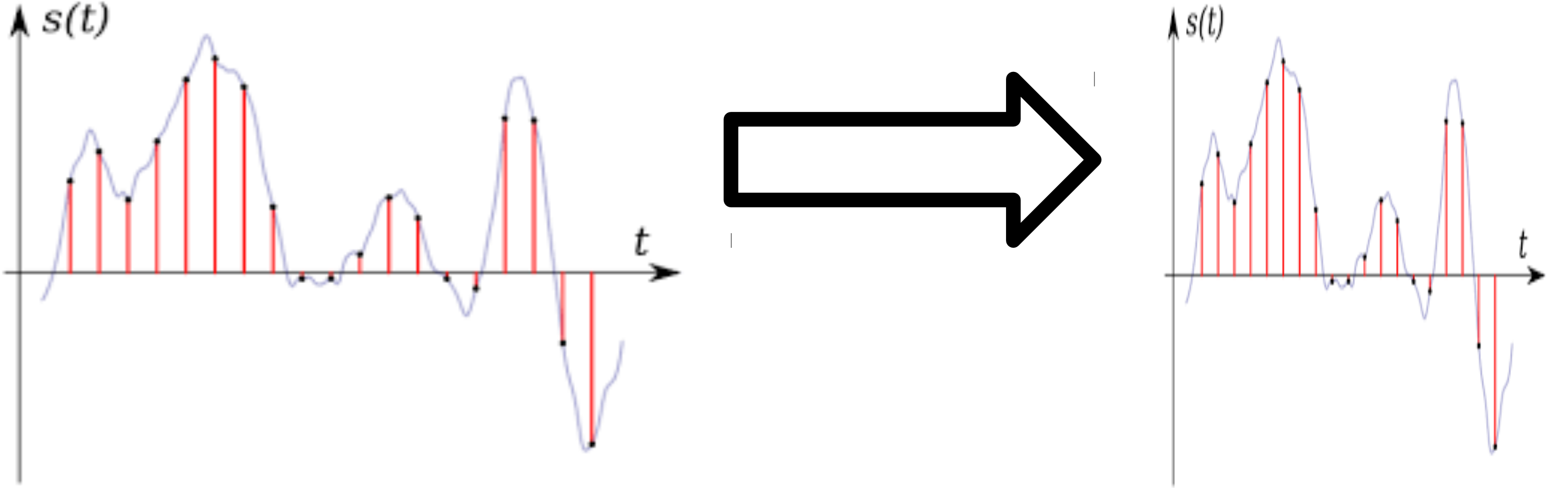
Review: Multimedia Similarity

Perceptual features for audio

Cool audio effects that aren’t LTI systems

Pitch shifting & time-stretching

Technique 1: Resample

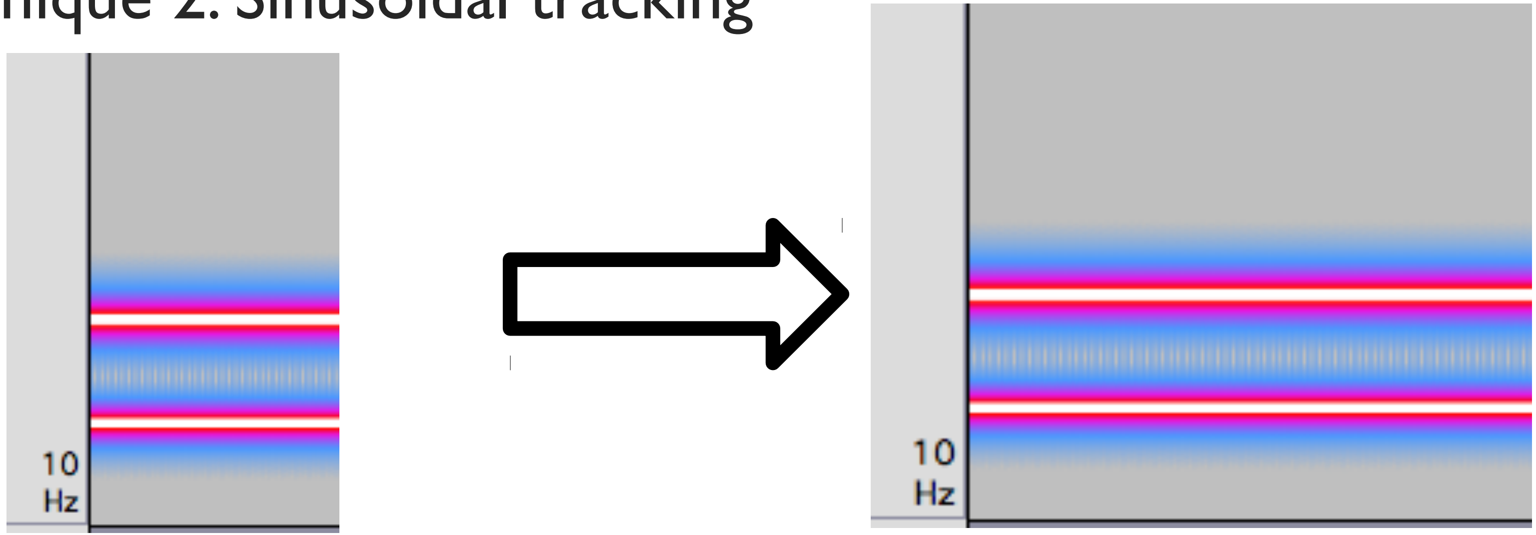


Problems?

```
file2 =  wavReadMono("~/audio/12345.wav")
play(file2, rate=80000)
```

Pitch shifting & time-stretching

Technique 2: Sinusoidal tracking

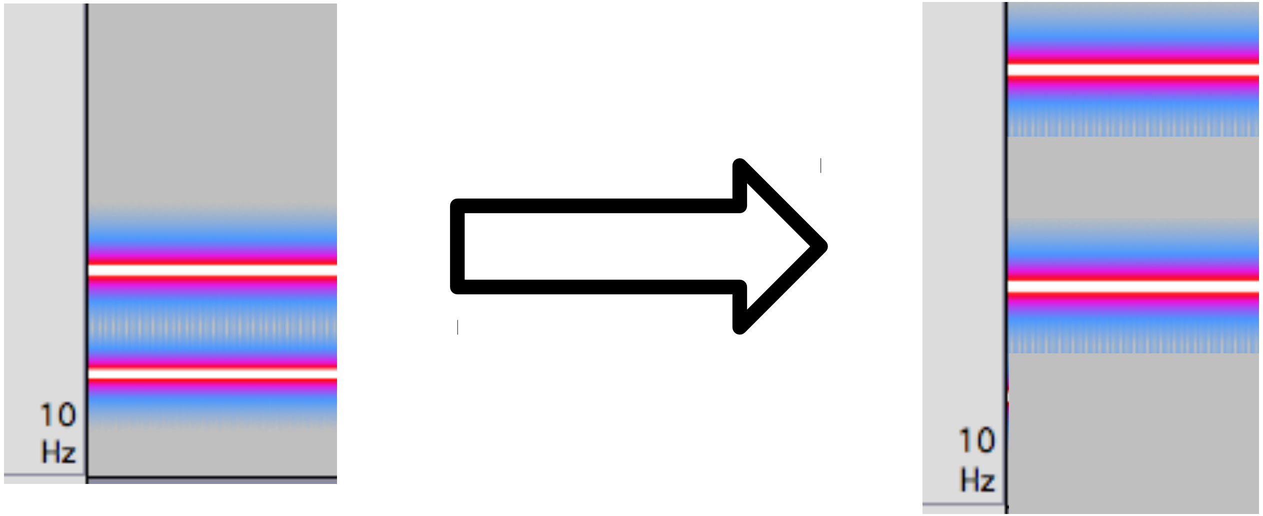


Use STFT to understand frequency content

Resynthesize a sound as sum of sine waves, control their amplitudes over time (e.g., at half the rate)

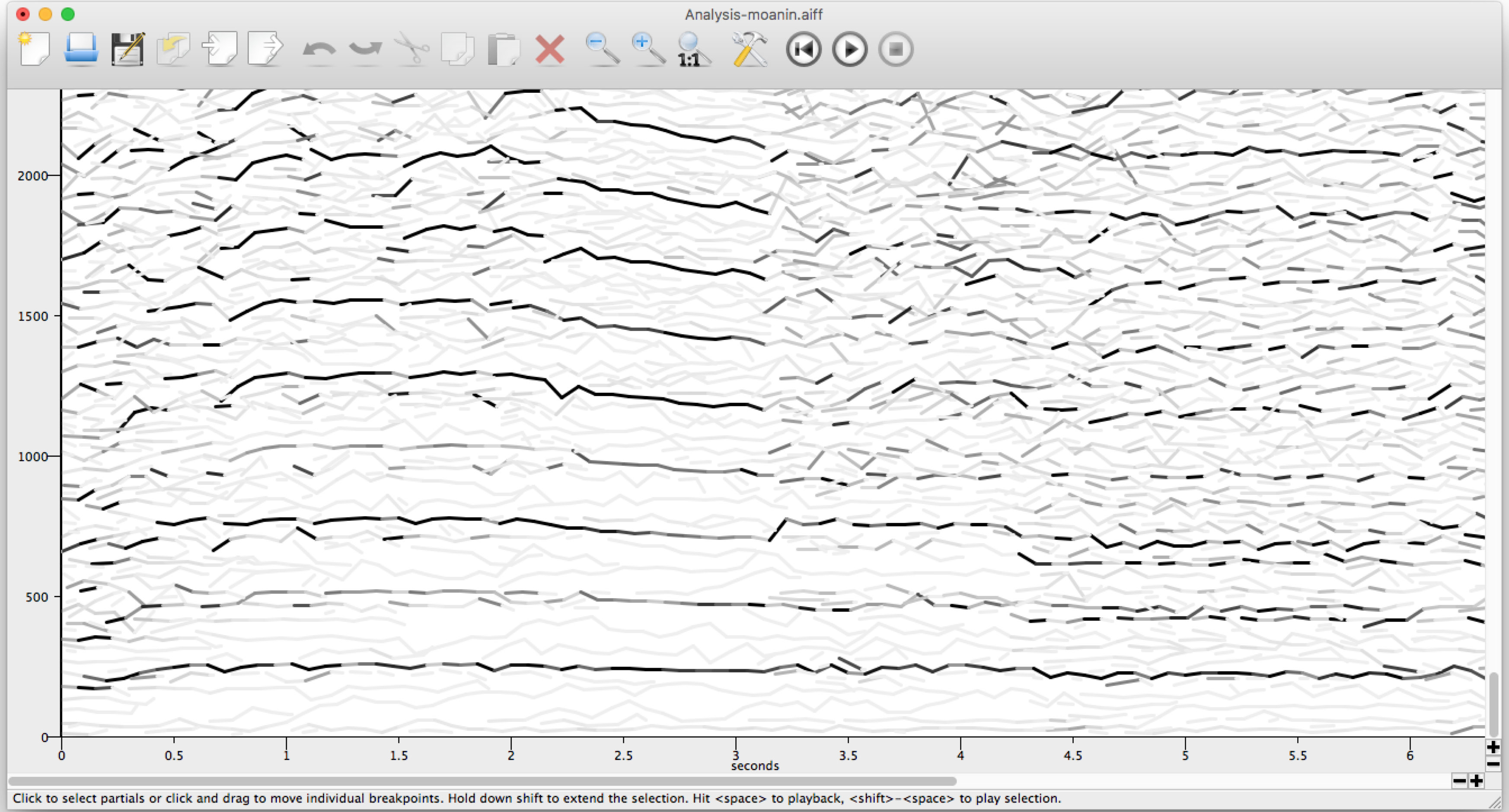
Pitch shifting & time-stretching

Technique 2: Sinusoidal tracking



Resynthesize a sound as sum of sine waves, change their frequencies (e.g., at twice the originals)

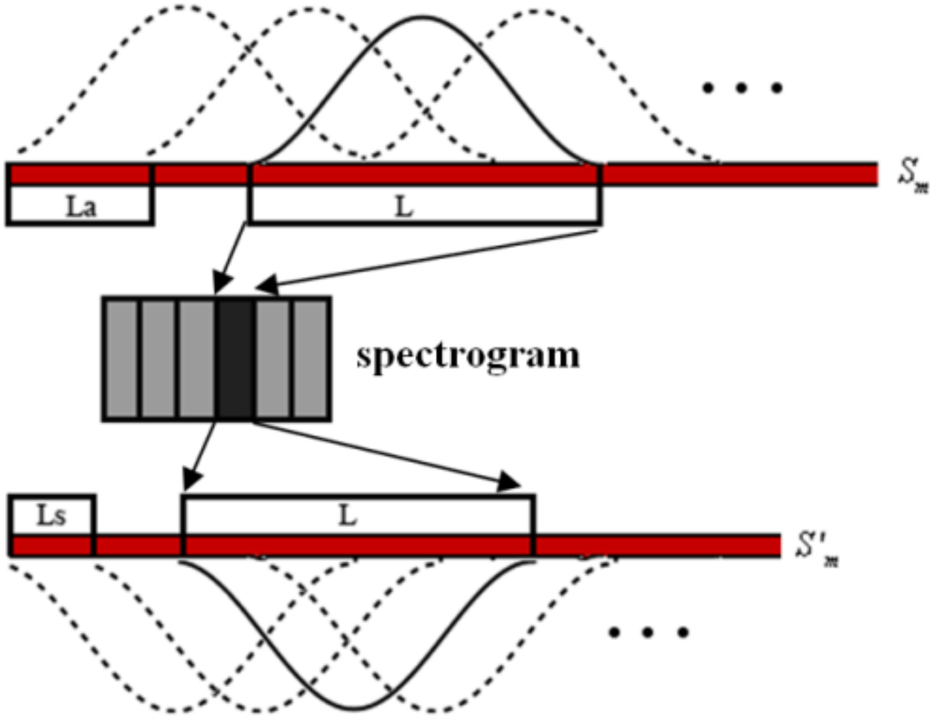
Problems?



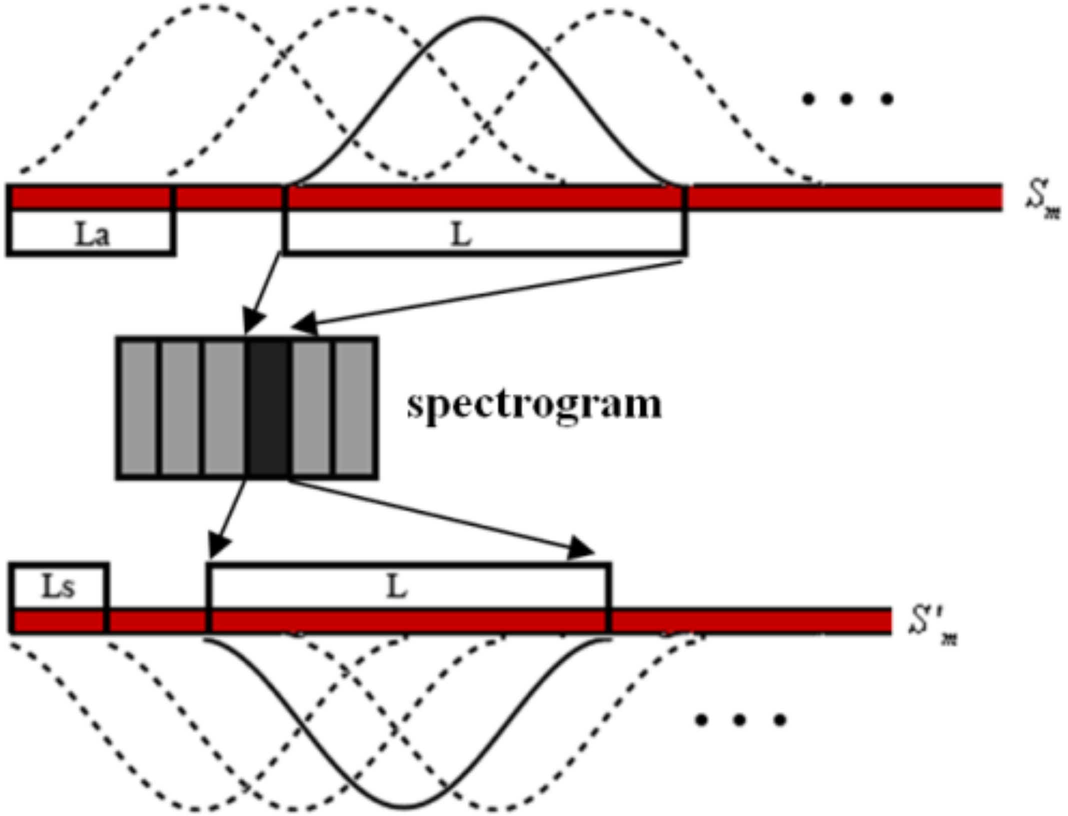
eg: http://www.klingbeil.com/spear/

Pitch shifting & time-stretching

Technique 3: “Overlap & add”



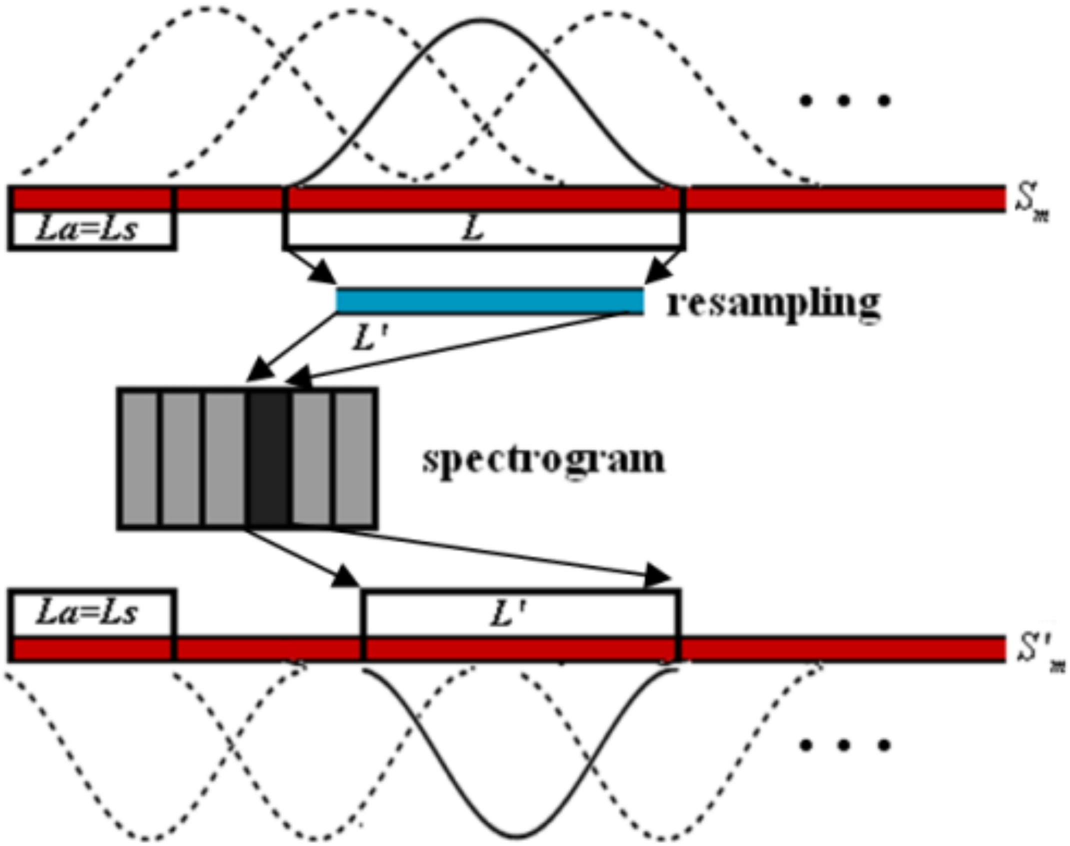
Time stretching w/ overlap-add



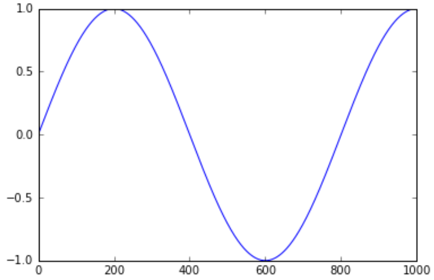
Use different hop size in input vs output

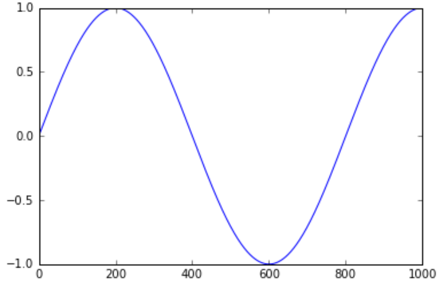
https://www.codeproject.com/Articles/245646/How-to-change-the-pitch-and-tempo-of-a-sound

Pitch shifting with overlap-add

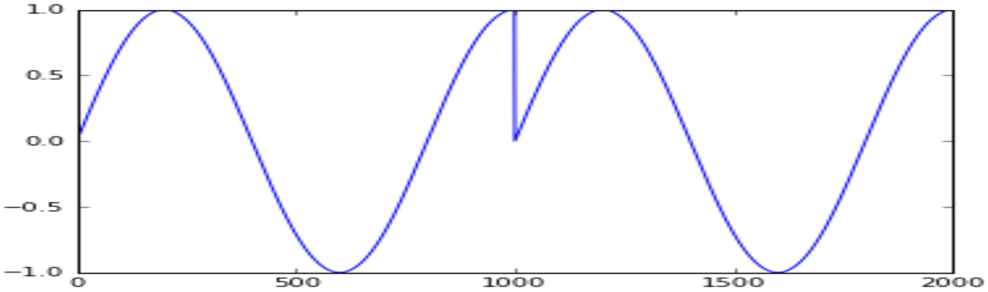


What happens when we add?

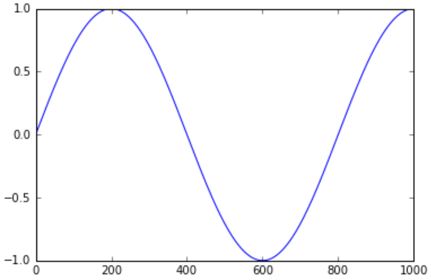


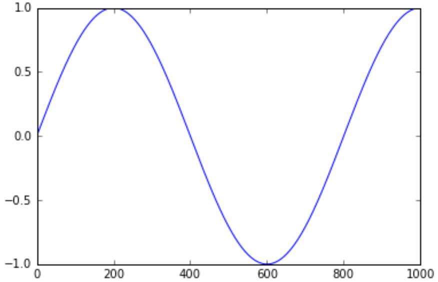


+

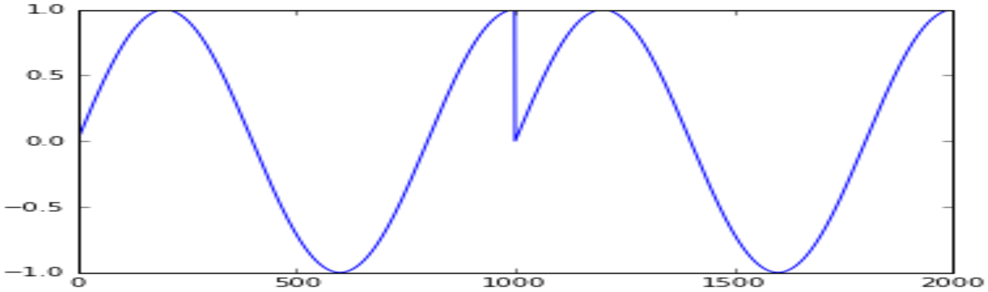


Better to keep track of phase...



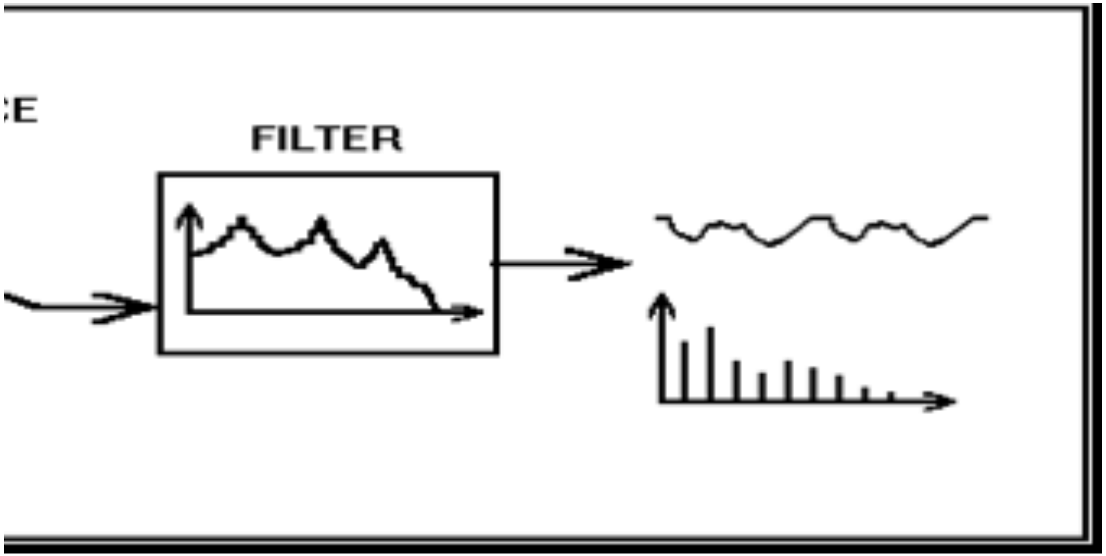


+



See:
Phase Vocoder

model & cross-synthesis



Demo:
guitar source spectrum shaped by voice spectrum

```
#cross-synthesis code
#first, load (or synthesize) sounds into file1 and file2

newSound = zeros(size(file1))
i = 0
j = 0;
width = 2048;
hop = 2048/16;
win = np.hamming(2048)

while (i+ width < size(file1)) :
    if (j + width >= size(file2)) :
        j = 0
    frame1 = win*file1[i:i+width]
    frame2 = win*file2[j:j+width]
    f1 = fft.fft(frame1)
    f2 = fft.fft(frame2)
    ms = abs(f2)
    frame3 = f1 * ms

    sig1 = fft.ifft(frame3).real
    newSound[i:i+width] = newSound[i:i+width] + sig1
    i = i + hop
    j = j + hop

newSound = 0.7*newSound/(max(abs(newSound)))
play(newSound)
plot(newSound)
```

To read more

Pitch shifting & time stretching:
http://blogs.zynaptiq.com/bernsee/time-pitch-overview/

Description of signal processing in the phase vocoder:
https://www.eumus.edu.uy/eme/ensenanza/electivas/dsp/presentaciones/PhaseVocoderTutorial.pdf

Third-year module info

Computer “Listening”

“What does this music sound like?”

What note is playing?

What chord is playing?

When do notes happen?

What key is it in?

What tempo is it?

What genre is it?

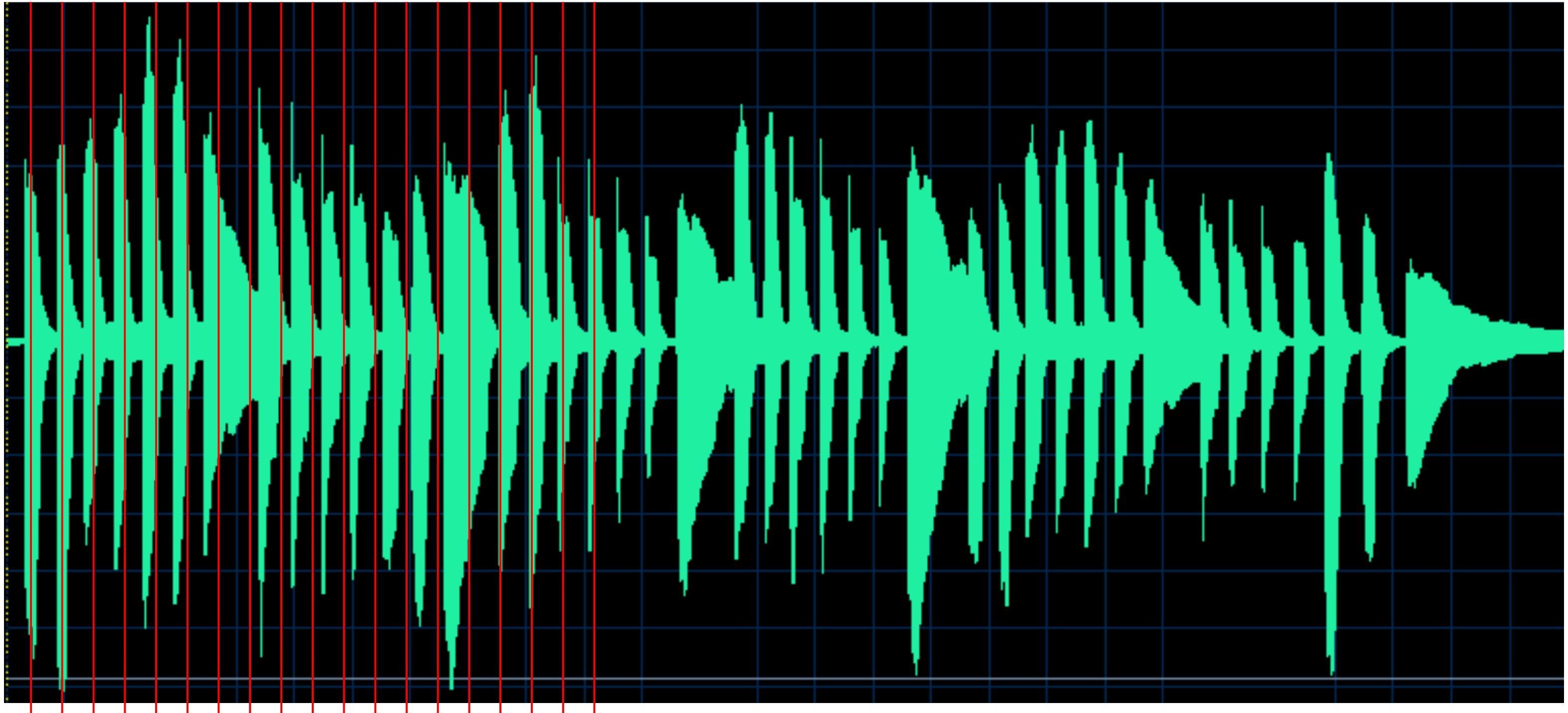
Would I like it?

Computing audio features

Feature: compute a value (or vector of values) from a sound

Sometimes a feature is useful enough on its own

Sometimes a "feature" tells us everything we need to know...



Computing audio features

Feature: compute a value (or vector of values) from a sound

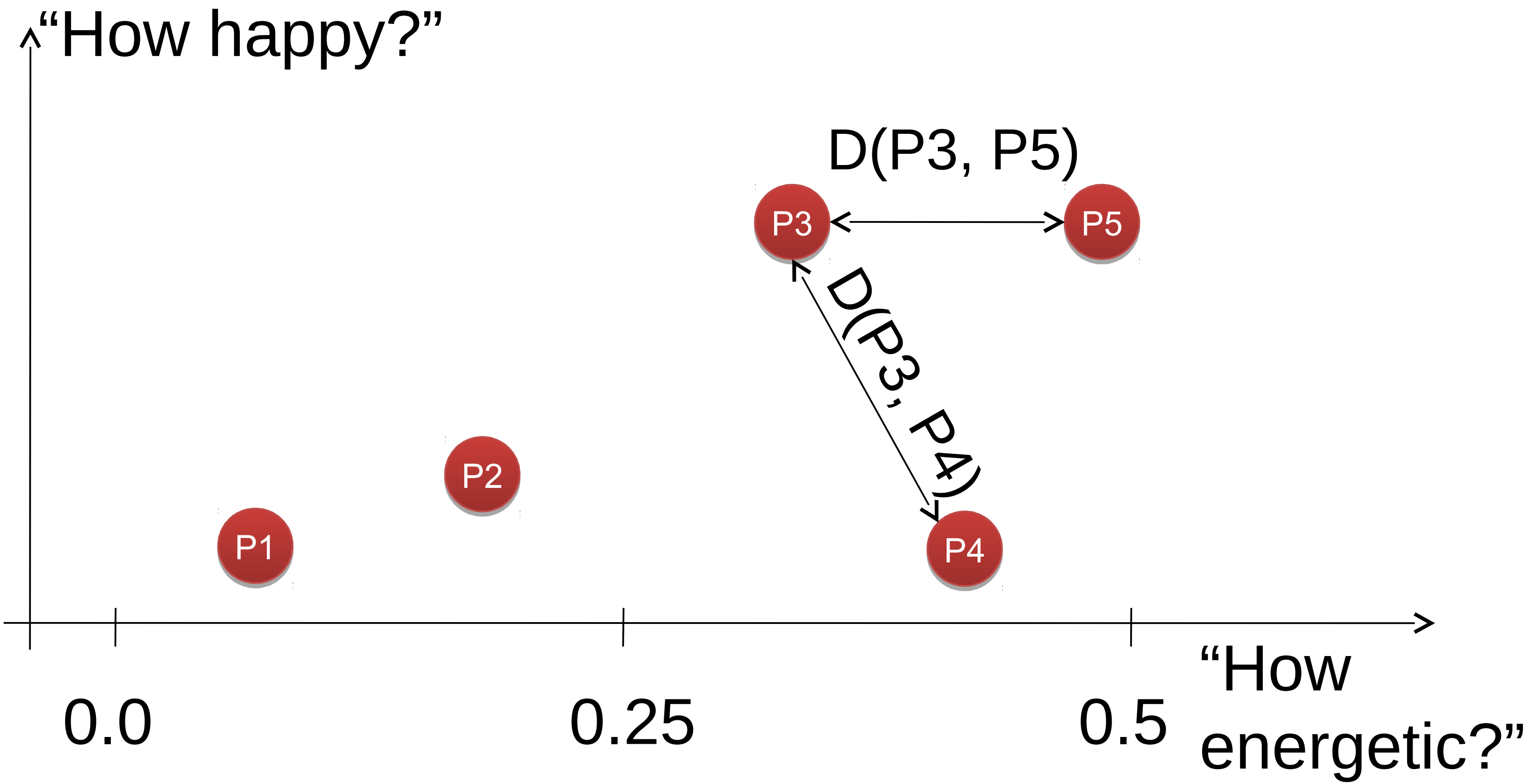
Sometimes a feature is useful enough on its own

Or you can compare it with other features:

- To some "ideal" value (e.g., "how much is this like a C major chord?")
- To some other example sound
  - Measure *similarity* between sounds: For labeling, recommendation, search, ...

Or you can use machine learning...

Euclidean distance: "As the crow flies"



Features & similarity

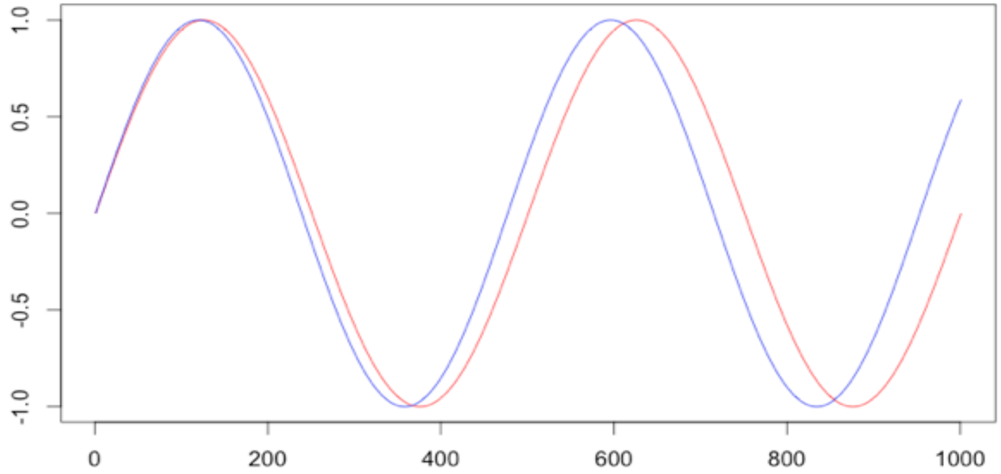
In general:

If two sounds are perceptually similar, we'd like them to have similar values for features

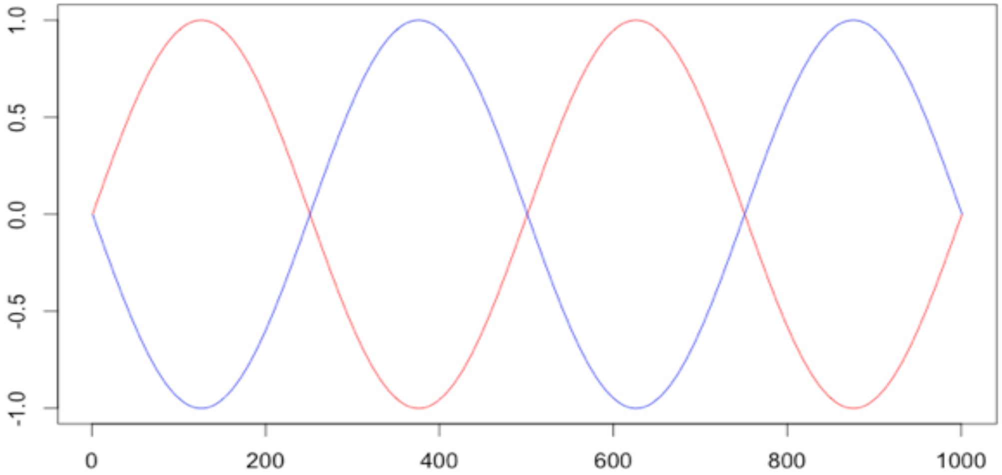
E.g., Sound 1 and Sound 2 have almost the same pitch: Let's find a feature /features for describing pitch which give them similar values.

Commonly Used Perceptual Features for Music and Sound

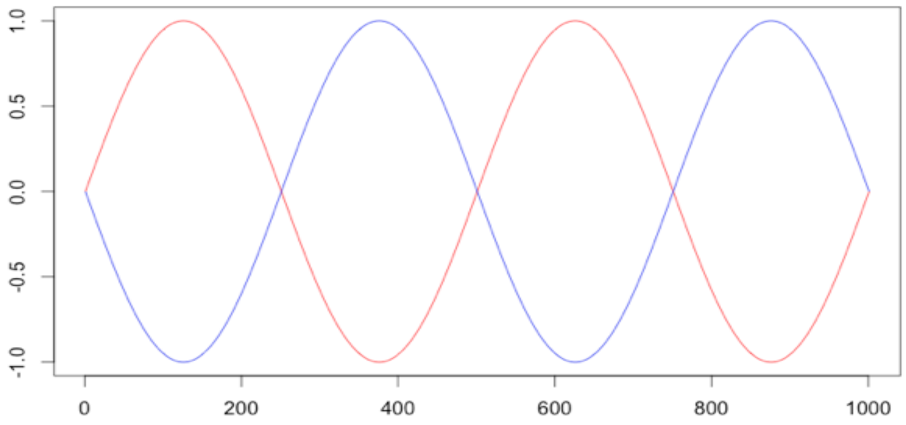
Simple audio features



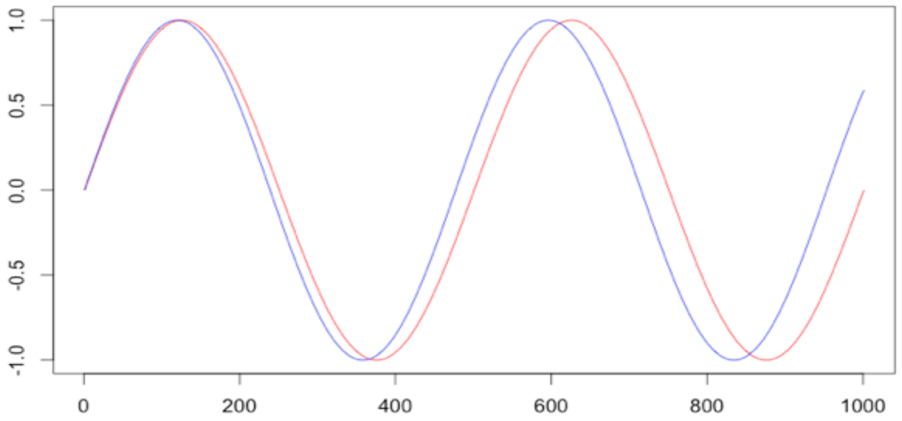
How similar will these sound?



How similar will these sound?



Sound exactly alike!
Euclidean distance
= 44.72



Sound different!
Euclidean distance
= 8.34

Computing distance in Python: `dist(s1, s2)`

☹

Big problem: Two sounds can sound *exactly* alike but have *very* different waveforms!

You should NOT use samples of the waveform as your features!

Another possibility:

- If waveform is N samples, then take its FFT, and use the i^{th} FFT bin as the i^{th} feature

- Often better than using samples! However, still problems:
  - hard to reason about how changes in pitch, timbre, volume, instrumentation, etc. will produce relative changes in distance
  - Overly sensitive to small changes in high frequency content

- There is probably a better feature representation that more precisely captures similarity for a given application…

Review: Loudness

- Loudness: related to *perceived* energy or power in audio
- Typically expressed in dB

$$10 \log_{10}\left(\frac{P_1}{P_0}\right) \text{ dB}$$

where P_0 is power @ threshold of hearing (0 dB).

- Increase of 10(?) dB roughly corresponds to doubling of loudness

http://www.indiana.edu/~emusic/acoustics/amplitude.htm

Simple features for loudness

- Do *not* use direct measures of amplitude as a feature ("peak amplitude" or "average amplitude")

  - our perception of loudness relates logarithmically to amplitude, so differences like A_1-A_2 are not meaningful

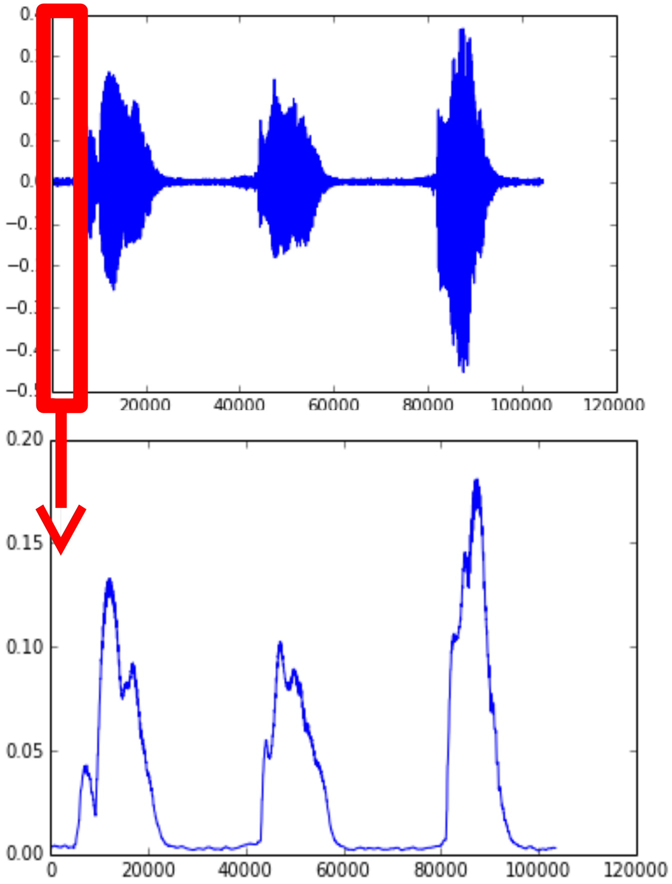
- Alternative: Use power spectrum (squared magnitude) expressed in dB (i.e., take the logarithm)

- Another common alternative: RMS ("root-mean-square") of audio waveform:

$$x_{\mathrm{rms}} = \sqrt{\frac{1}{n}\left(x_1^2 + x_2^2 + \cdots + x_n^2\right)}$$

RMS (root-mean-square)

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}$$



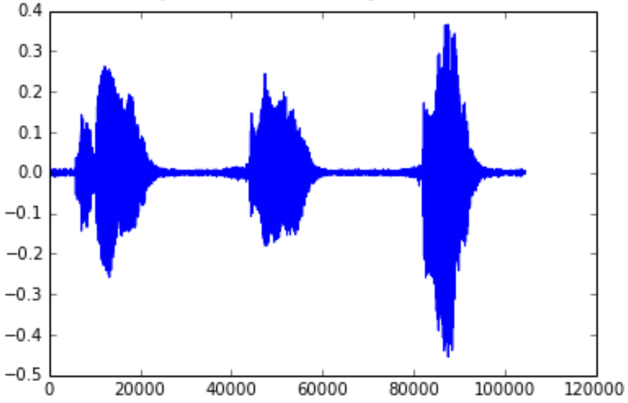
RMS of 1000-sample window, hop size 100 samples:

Reasonable for volume

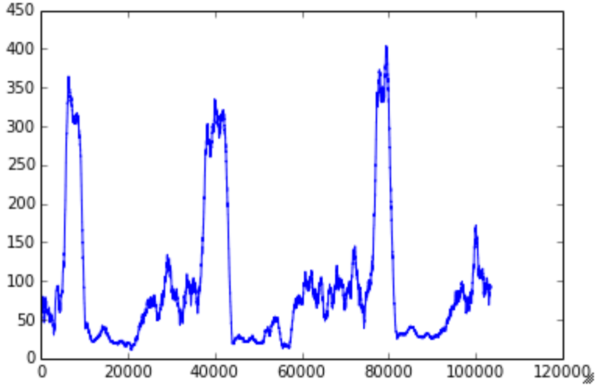
RMS demo: Using sndpeek

ZCR (zero-crossing rate)

How many times does the signal cross zero?



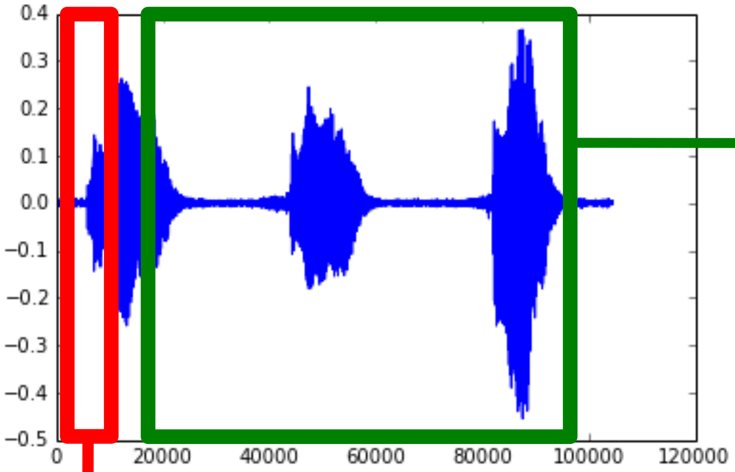
ZCR of 1000-sample window, hop size 100 samples:

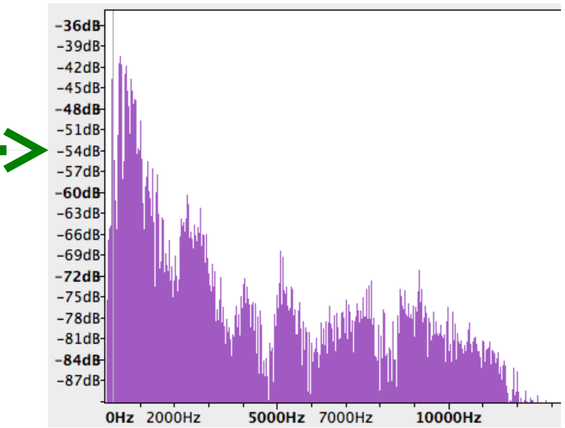


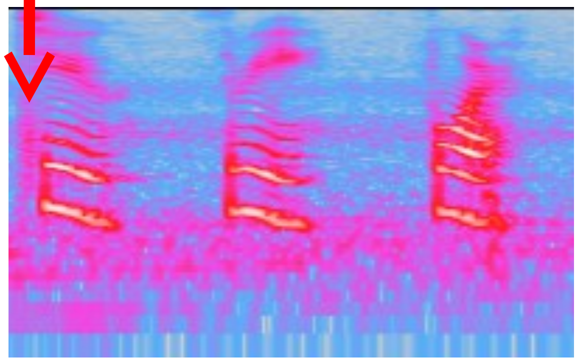
Reasonable for noisiness, silence, ~pitch

Spectral features

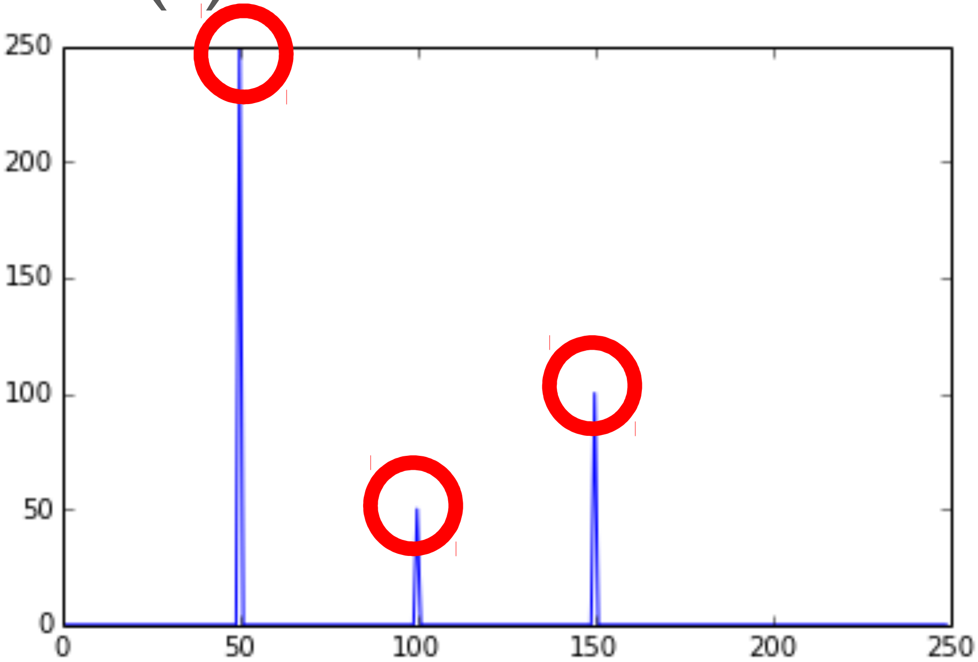
From the FFT or STFT of a signal:







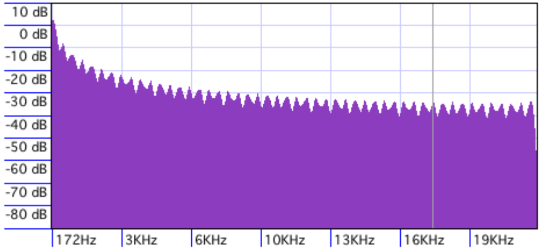
Spectral peak(s)



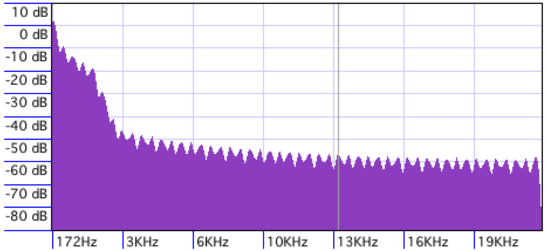
What are most prominent frequencies?
What is fundamental frequency?

Review: Timbre

- Timbre of a sound: related to several factors, one of which pertains to amount of high frequency content
  - More high frequency = “brighter”, less = “darker”



“brighter”

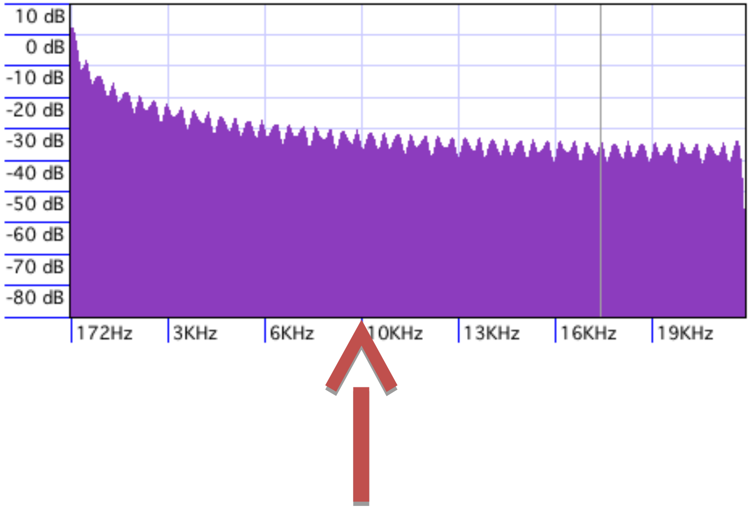


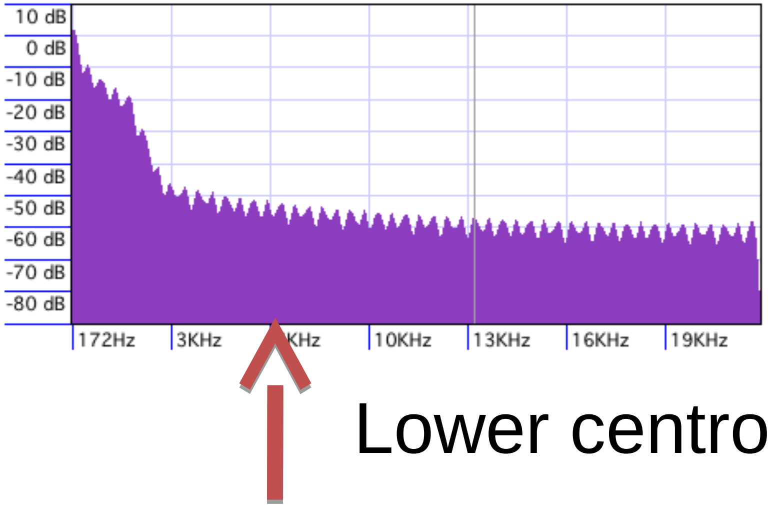
“darker”

A simple timbre feature

Spectral centroid: "center of mass" of the spectrum

$$\mathrm{SC}(m) = \frac{\sum_k f_k |X(m,k)|}{\sum_k |X(m,k)|}$$



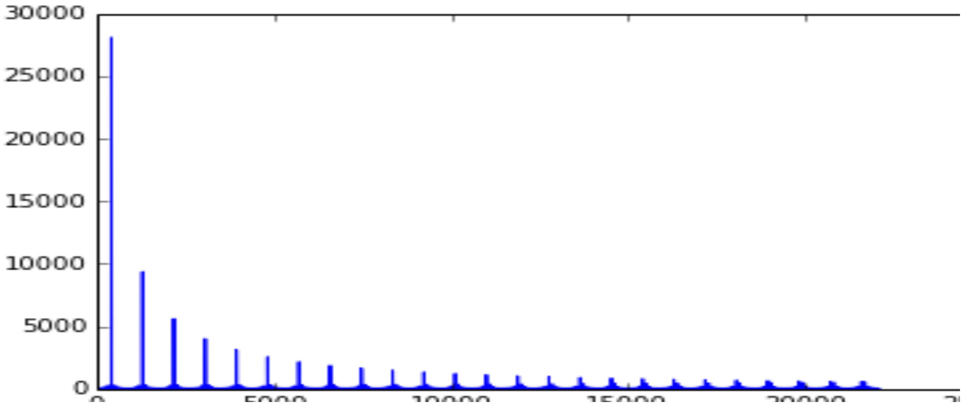


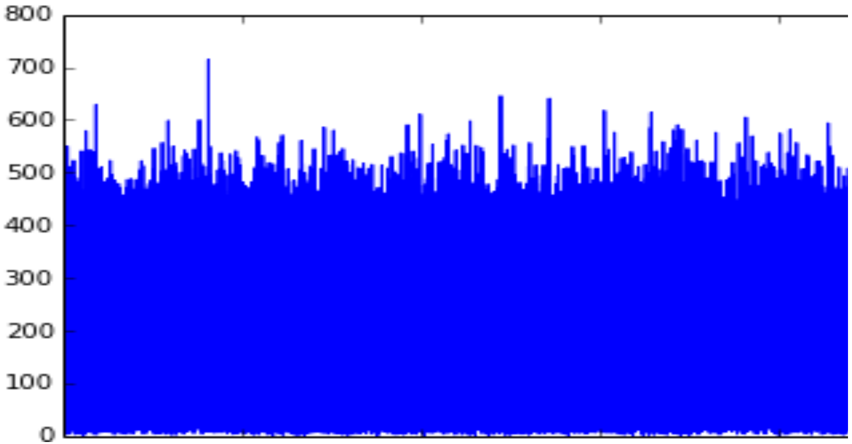
Lower centroid

Spectral centroid demo

Spectral flatness:

How “noise-like” vs. “tone-like”?





Tone-like

Noisy

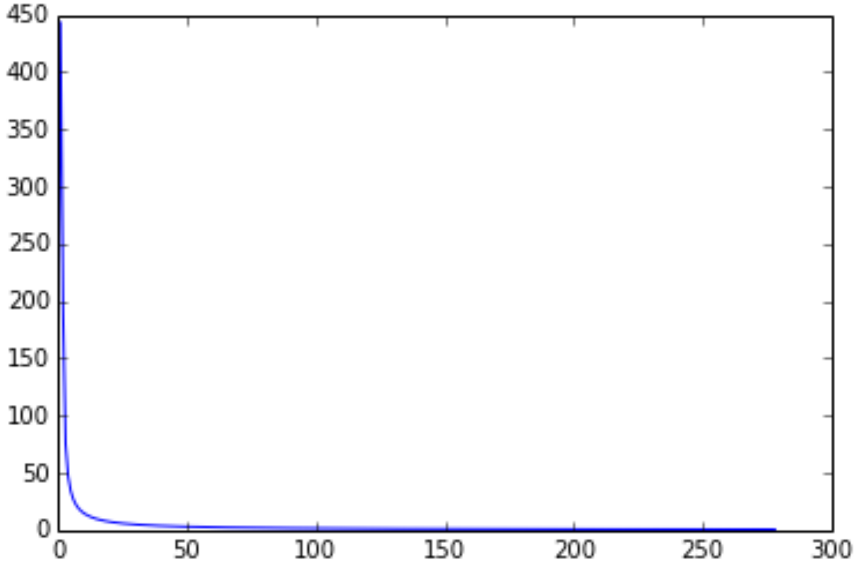
$$\text{Flatness} = \frac{\sqrt[N]{\prod_{n=0}^{N-1} x(n)}}{\frac{\sum_{n=0}^{N-1} x(n)}{N}} = \frac{\exp\left(\frac{1}{N}\sum_{n=0}^{N-1} \ln x(n)\right)}{\frac{1}{N}\sum_{n=0}^{N-1} x(n)}$$

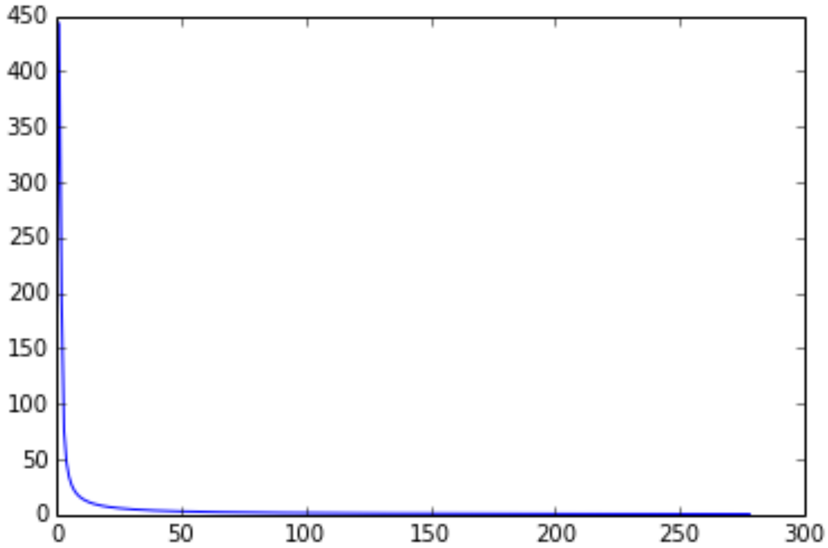
More complex audio features

Some problems with FFT/STFT

1. FFT has linear pitch scale: Perceptual differences between pitches don’t match differences in FFTs

100 & 120Hz

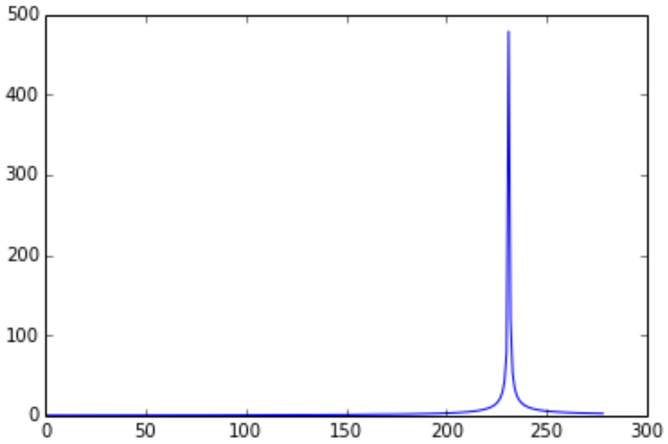


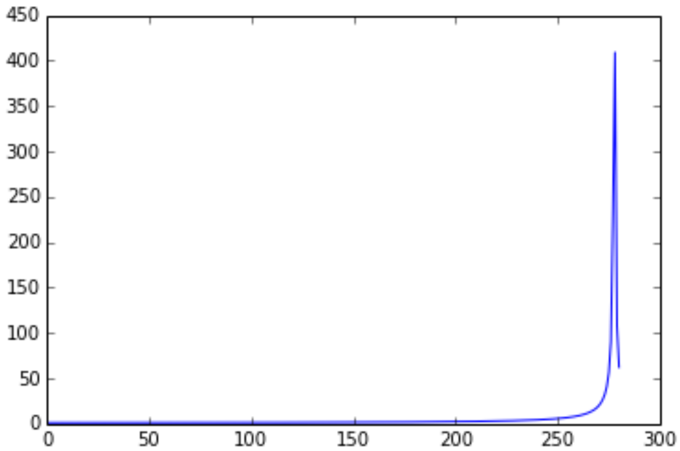


Some problems with FFT/STFT

1. FFT has linear pitch scale: Perceptual differences between pitches don't match differences in FFTs

1000 & 1200Hz





Some problems with FFT/STFT

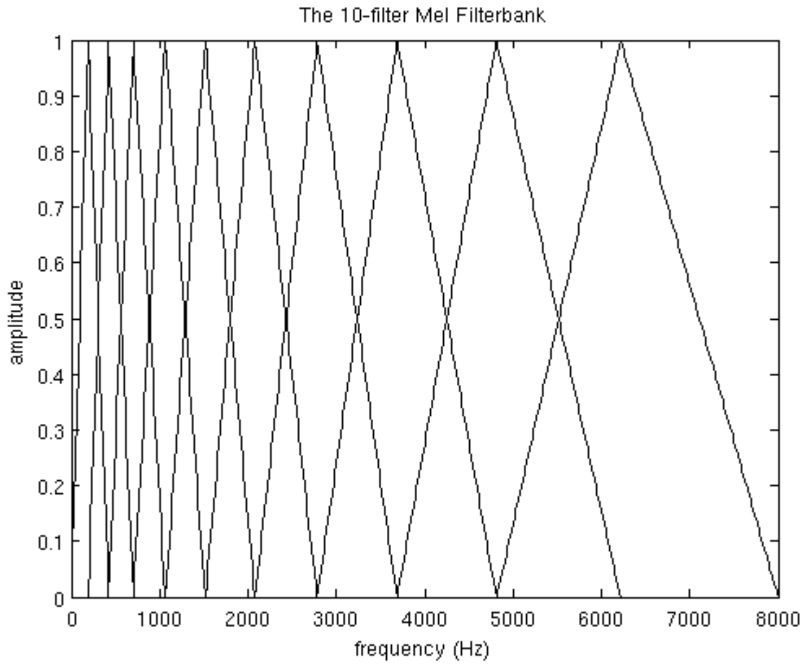
1. FFT has linear pitch scale: Perceptual differences between pitches don't match differences in FFTs

Solution: Transform FFT bins into perceptually-spaced bins (or use a slightly different transform)

2. Perceptual differences in volume are not accurately represented by differences in bin magnitudes

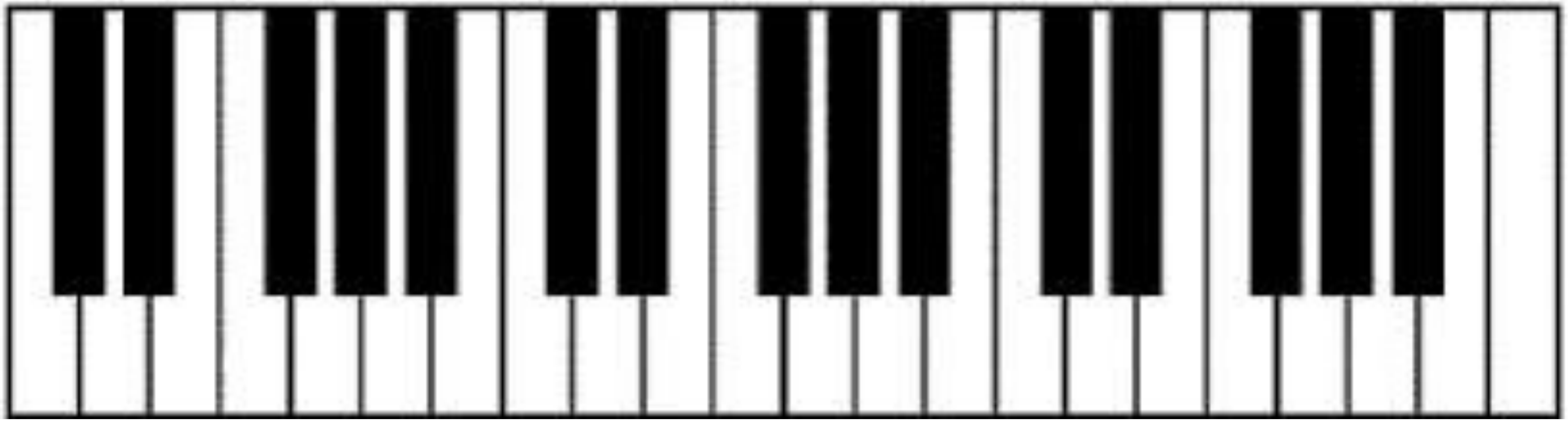
Simple solution: Take log of magnitude spectrum.
More complicated: Use bark scale coefficients

Mel scale



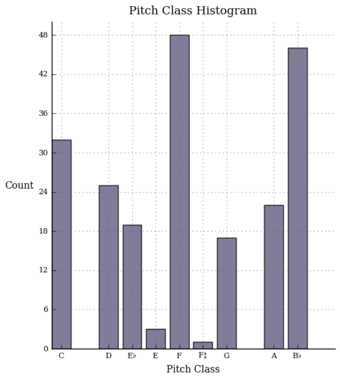
Can visualize mel coefficients
instead of FFT
bin magnitudes

Or, combine bins to make one bigger bin per pitch:



Constant-Q with semitone bands

Or, combine all pitches of the same chroma (note name, e.g. all As, all Bb, etc.:



“Pitch histogram”

One more feature: speech & timbre (& instrumentation, genre, …)

Mel Frequency Cepstral Coefficents (MFCCS)

MFCCs

1. Take FFT
2. Convert magnitudes to mel scale
3. Take log of mel scale bins
4. Compute Discrete Cosine Transform (like FFT)

Spectrum of a spectrum: Cepstrum

Why???

Log of mel bins: because we hear loudness logarithmically

DCT: Low-quefrency bins tell us about overall shape of mel spectrum

Yes, that says quefrency.

MFCC demo

More reading on MFCCs:

For good explanations:

- See slide 24 of http://eamusic.dartmouth.edu/~mcasey/m103/m103_8.pdf
- Also http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/

Final comments on spectral features

Typically computed from STFT or equivalent (sliding analysis window)

May have many feature values per audio file/song

- This can help analyze song: e.g., find boundaries between verse/chorus, or periods of silence

- If doing analysis of a whole song, combine these in some way (e.g., take mean and standard deviation of 1st, 2nd, 3rd cepstral coefficients; this becomes new feature vector)

Example feature representations for real applications

Speech recognition

Window size 16ms-25ms, 50% overlap

Compute MFCCs

Use only first 13 MFCCs per window

Musical genre classification

- 100ms windows, 50% overlap
- Constant-Q spectrum with 12 bins per octave
- Add powers (ignoring subtle volume effects)
- Compute cepstrum, keeping first 20 cepstral coefficients

Cover song detection

- 500ms windows, 40% overlap
- Constant-Q spectrum with 12 bins per octave
- Octave folding (by adding powers)

How do I choose features?

- Depends on the context!
- Often, many features might work
- Use your judgment and knowledge of perception & task
- Experiment!

For more information

http://www.nyu.edu/classes/bello/MIR_files/timbre.pdf

http://earsketch.gatech.edu/learning/teaching-computers-to-listen-2-analysis-features

Audio feature extraction & visualization tools

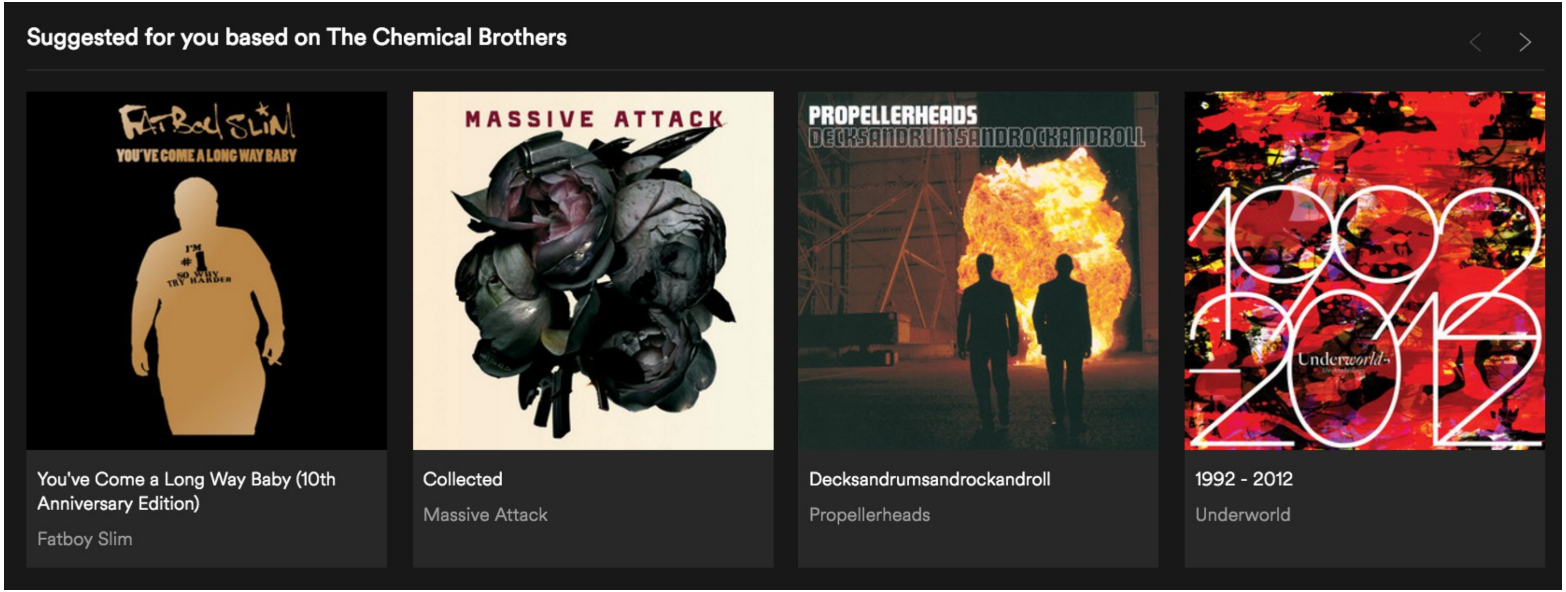
Sonic Visualizer: Plug-ins for these audio features and MANY MORE

jAudio: free and cross-platform toolkit for extracting features from audio recordings

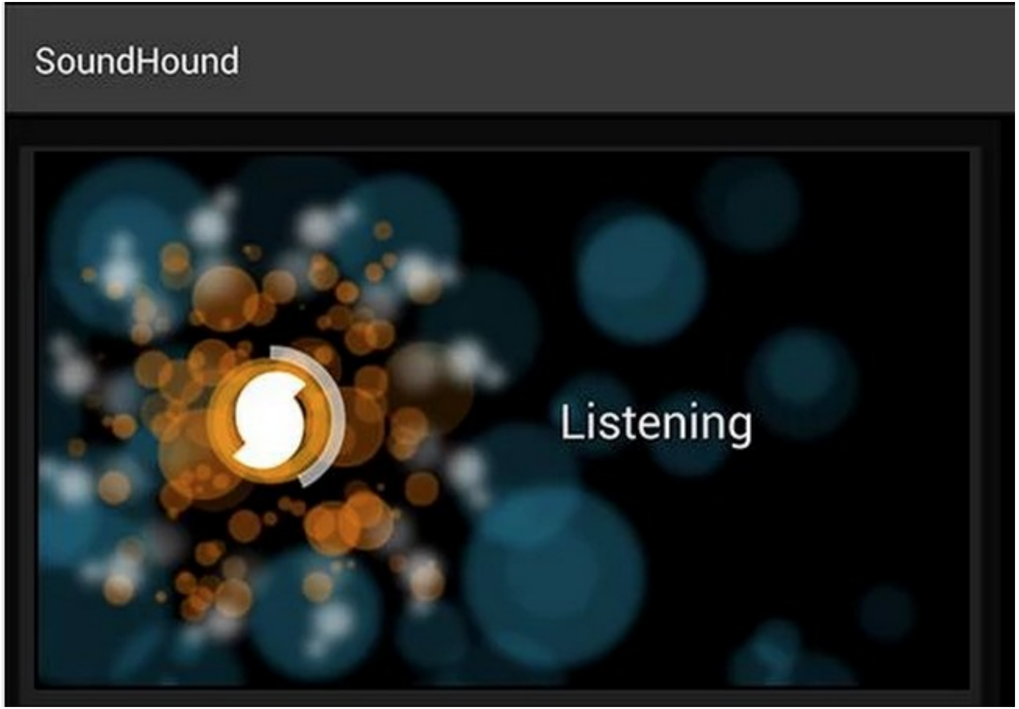
Example datasets

- Million song dataset:
  - Features already extracted (includes MFCC-derived features and meta-data features)
  - http://labrosa.ee.columbia.edu/millionsong/pages/example-track-descriptiona
- Audio & symbolic datasets commonly used in music IR:
  - http://grh.mur.at/sites/default/files/mir_datasets_0.html

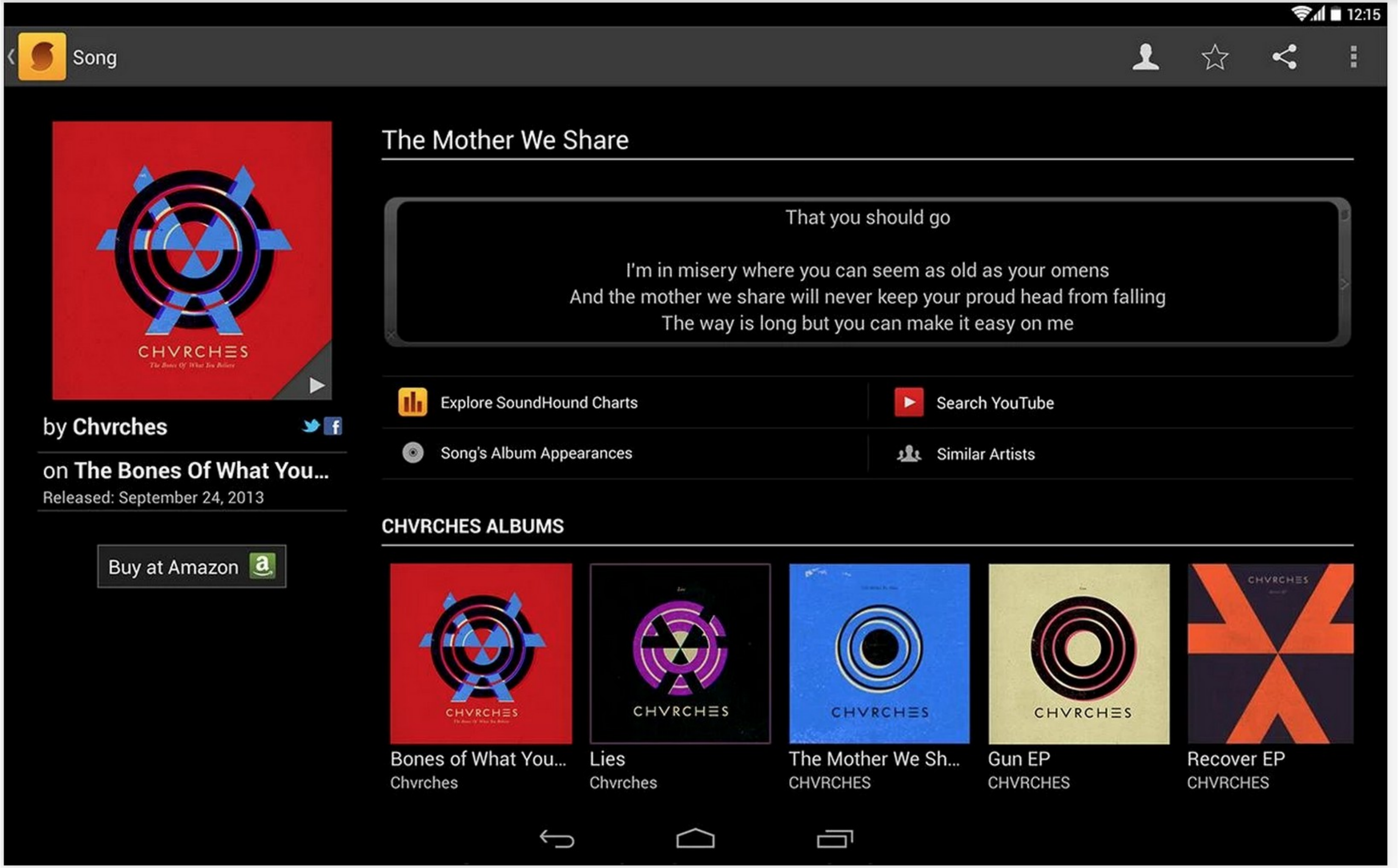
Example: Spotify

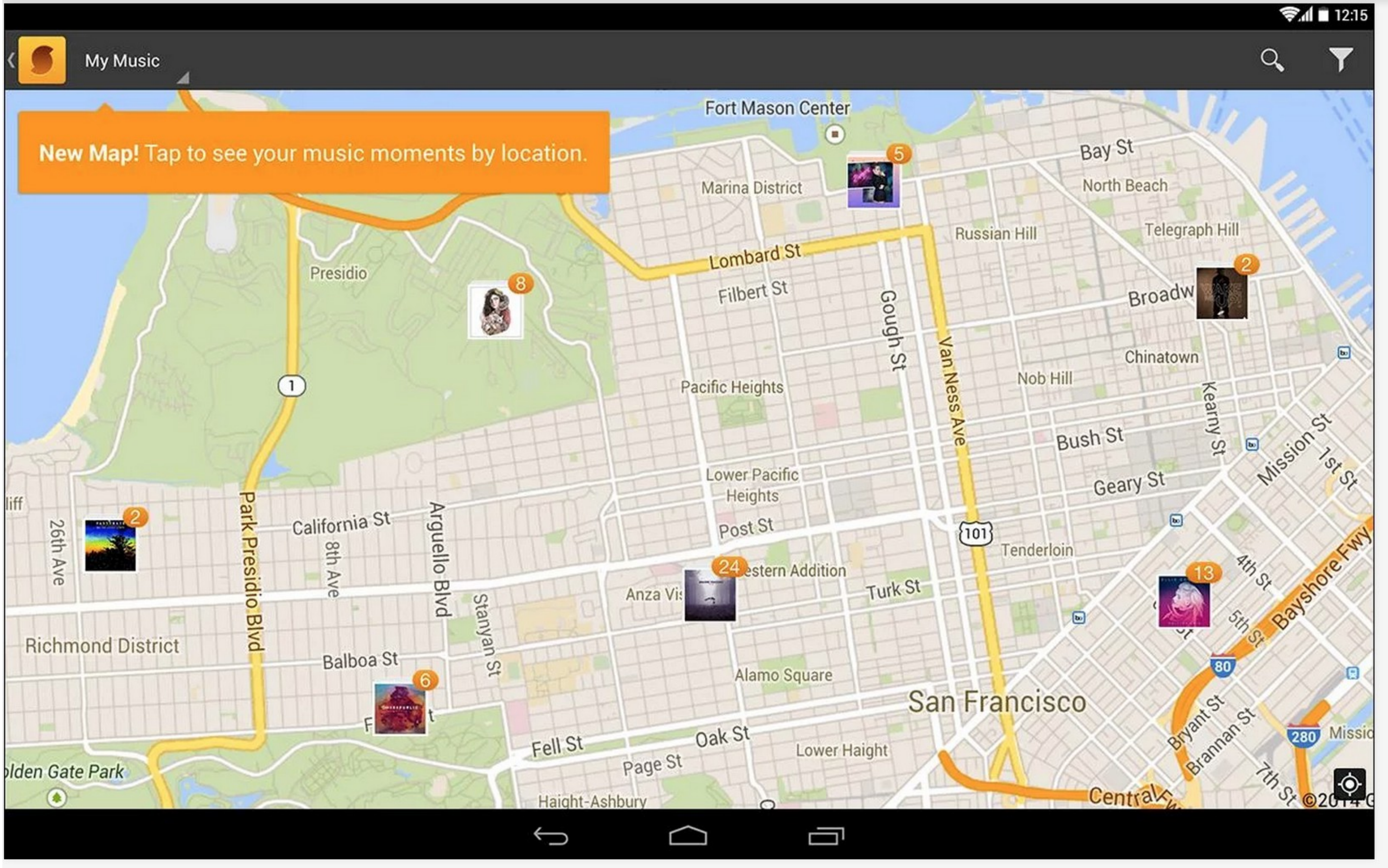


Example: SoundHound



“What song is playing?”


12:15
Song
The Mother We Share
That you should go
I'm in misery where you can seem as old as your omens
And the mother we share will never keep your proud head from falling
The way is long but you can make it easy on me
Explore SoundHound Charts
Search YouTube
Song's Album Appearances
Similar Artists
CHVRCHES
by Chvrches
on The Bones Of What You...
Released: September 24, 2013
Buy at Amazon
CHVRCHES ALBUMS
CHVRCHES
CHVRCHES
CHVRCHES
CHVRCHES
CHVRCHES
Bones of What You...
Chvrches
Lies
Chvrches
The Mother We Sh...
CHVRCHES
Gun EP
CHVRCHES
Recover EP
CHVRCHES



What are people listening to nearby?