MIR démo

MIRtoolbox

General Principles

List of all the functions available in *MIRtoolbox*:

```
help mirtoolbox
```

Help for one specific function:

```
help miraudio
```

Display the waveform of audio file ‘ragtime.wav’ and store the result in a variable a:

```
a = miraudio(‘ragtime.wav’)
```

Add ‘;’ to avoid the display of the results:

```
a = miraudio(‘ragtime.wav’);
```

Optional operations such as centering and resampling:

```
a = miraudio(‘ragtime.wav’, ‘Center’, ‘Sampling’, 11025)
```

or in several steps:

```
a = miraudio(‘ragtime.wav’, ‘Center’)
a = miraudio(a, ‘Sampling’, 11025)
```

Play the resulting audio:

```
mirplay(a)
```

Analyze all files of the current folder (for instance, the folder located at the address: MIRToolboxDemos/train_set) :

```
miraudio(‘Folder’)
```

Compute for instance the spectrum of a:

```
mirspectrum(a)
```

Obtain the output in numeric format:

```
mirgetdata(a)
```

Basic Operators

Trim the silence at the beginning and the end of the audio:

```
a = miraudio(a,’Trim’)
```

Extract the first second only:

```
miraudio(a,’Extract’,0,1)
```

Extract the amplitude envelope:

```
e = mirenvelope(a)
```

Compute the spectrum (Fourier transform):

```
s = mirspectrum(‘Amin3.wav’)
```

Select frequencies below 3000 Hz:

```
s = mirspectrum(s,‘Max’,3000)
```

Display the energy in decibel scale:

```
s = mirspectrum(s,‘dB’)
```

Decompose the energy in Mel bands:

```
s = mirspectrum(a,'Mel')
```

Detect peaks:

```
mirpeaks(s)
```

Decompose the audio into frames of length 0.1 s. and half-overlapped:

```
f = mirframe('ragtime.wav',.1,.5)
```

Compute the spectrum for each frame (i.e., spectrogram):

```
s = mirspectrum(f)
```

This can be written in one line:

```
s = mirspectrum('ragtime.wav', 'Frame', .1, .5)
```

Detect the peaks:

```
mirpeaks(s)
```

Compute the resulting spectral flux:

```
mirflux(s)
```

The "flux" can be computed for any frame-decomposed representation, for instance:

```
mirflux(mirautocor(a,'Frame'))
```

Decompose the audio signal into 5 channels:

```
fb = mirfilterbank(a, 'NbChannels', 5)
```

Compute the envelope in each channel:

```
e = mirenvelope(fb)
```

Compute the autocorrelation function for each envelope in each channel:

```
ae  = mirautocor(e)
```

Compute the autocorrelation summary:

```
sa = mirsum(ae)
```

Feature Extractors

Dynamics

Root-mean-square energy curve:

```
r1 = mirrms('movie1.wav', 'Frame')
r2 = mirrms('movie2.wav', 'Frame')
```

Low-energy rate:

```
mirlowenergy(r1)
mirlowenergy(r2)
```

Rhythm

Tempo estimation:

```
[t,ac] = mirtempo(‘ragtime.wav’)
```

Temporal evolution of tempo estimation:

```
[t,ac] = mirtempo('czardas.wav', ‘Frame’)
```

Timbre

Attack Onset estimation:

```
mironsets(‘ragtime.wav’)
```

Attack slope:

```
mirattackslope(‘ragtime.wav’)
```

Brightness

```
mirbrightness(‘ragtime.wav’, ‘Frame’)
```

Roughness:

```
mirroughness(‘ragtime.wav’, ‘Frame’)
```

Pitch and Tonality

Multi-pitch extraction:

```
[p,a] = mirpitch(‘ragtime.wav’, ’Frame’,’Multi’)
```

Wrapped chromagram:

```
mirchromagram(‘ragtime.wav’)
```

Tonality estimation:

```
mirkey(‘ragtime.wav’)
k = mirkey(‘ragtime.wav’,’Frame’)
```

Mode estimation:

```
mirmode(‘ragtime.wav’)
m = mirmode(‘ragtime.wav’,‘Frame’)
```

rhythm and meter

Aims: To get familiar with tempo estimation from audio using the MIRtoolbox. To assess the performance of the tempo estimation method.

Part 1. Read the demo sounds ‘ragtime.wav’, 'vivaldi.wav’, 'valse_triste_happy.wav', 'laksin.wav' and 'czardas.wav' using the miraudio command. Estimate the tempos of the files in the folder. For this, go to the URL

```
http://www.metronomeonline.com/
```

where you can find an online metronome.

Play each excerpt using the mirplay command

```
mirplay('ragtime.wav');
```

Adjust the speed of the metronome to match with the tempo of the music. Write down each tempo. If the tempo varies within the excerpt (as is the case with 'czardas.wav' and "laksin.wav'), try to estimate the range of tempi within the excerpt.

Part 2. Use the function *mirtempo* to computationally estimate the tempi of the excerpts.

Compare the computational estimates with your perceptual estimates. To what extent do they agree? What are the typical discrepancies? Can you explain the reason for them?

Part 3. The excerpts 'czardas.wav' and 'laksin.wav' have variable tempi.

Use frame-based tempo analysis to estimate the time course of development tempo. To this end, decompose the two excerpts using the 'Frame' option of mirtempo (see help mirtempo; use a frame length of at least 2 s). What are the ranges of variation of tempi? Do they correspond to your estimates?

Part 4. Listen to the files in the folder *Pulse* and order them in terms of ascending tempo and ascending clarity of pulse.

Use the batch processing capability of the MIR Toolbox to calculate tempo and pulseclarity descriptors for a folder of sounds. For instance, calculate the tempo with the commands

```
t = mirtempo('Folder')
```

Does the resulting order match with your perceptual order? What are the discrepancies?

You can view the evolution of tempo of all the files using a frame-based approach by typing

```
t = mirtempo(‘Folder’,’Frame’,3,1,s)
```

What can you infer from the resulting figures when you type t ? Which audio files have a more steady tempo? What would that imply in terms of pulse clarity?

Estimate the Pulse Clarity of each of the songs and check if the output of the function *mirpulseclarity* matches your perceptual order.

```
ex: mirpulseclarity('Derezzed.mp3')
```

Do you agree with the ordering of the files? If not then explain possible reasons for the discrepancies?

repetition in music

For the songs available on the following link:

https://iiithydresearch-my.sharepoint.com/:f:/g/personal/lalit_mo_research_iiit_ac_in/EgK_5aRRstFKuKQggY0orXoBhqtT4ghvPLObw2sbwTjROg?e=HL8Ep7

For both the files:

1. the Chromagram vectors of this excerpt frame by frame, using the *mirchromagram* operator and the *'Frame'* keyword. Compute the similarity matrix of the sequence of chromagram, using the *mirsimatrix* operator.

2. repeat Step 1 using *mirspectrum* (instead of *mirchromagram)*.

Vary the frame length (250ms, 3s, 10s) and observe the change in granularity of the structure.

Which window length best represents your notion of perceptual segmentation and repetition?

timbre

Aims: To get familiar with the most common timbral descriptors used in MIR, and their perceptual meaning.

Use the batch processing capability of the MIR Toolbox to calculate timbral descriptors for the folder (*Stimuli*) of sounds. For instance, calculate the zero-crossing rates with the commands

```
zcr=mirzerocross('Folder')
```

You can now play all the files in the order of increasing zero-crossing rate by typing

```
mirplay(‘Folder’, ’Increasing’,zcr)
```

To save time, you can play only every 10th file by typing:

```
mirplay(‘Folder’, ’Increasing’,zcr, ‘Every’, 10)
```

Repeat the analysis with the spectral descriptors

low energy

```
mirlowenergy(‘Folder’)
```

spectral centroid

```
mircentroid(‘Folder’)
```

spectral rolloff

```
mirrolloff(‘Folder’)
```

spectral irregularity:

```
mirregularity(‘Folder’)
```

spectral entropy

```
mirentropy(‘Folder’)
```

Can you identify the perceptual correlates of spectral centroid, spectral rolloff and spectral entropy?

Calculate the correlation between these features.