

Ircamdescriptor (Matlab) documentation

System: Matlab 7.0
Version: 0.4
Date: 18/4/2008
Authors: peeters@ircam.fr, burred@ircam.fr
Copyright: ircam 2008

Presentation

Ircamdescriptor is a matlab p-file (pseudocode file) which runs under Matlab version 7 or above. The main reason for the use of p-file is to ensure no modification of the main code (which usually arise when providing direct matlab source code), and to ensure cross-platform compatibility (the p-file runs either under Linux, Windows or Mac-OS). Ircamdescriptor is a single file that contains the whole code for reading files, computation of descriptors and writing.

Comments on the use of harmonic/noise features

For the computation of all the features that are based on either the FundamentalFrequency or the Harmonic modeling of the sounds, (such as OddEvenHarmonicRatio), you need to provide the program with the F0 and Partial analysis SDIF files. In this case, an external software is needed in order to compute the harmonic sinusoidal representation of a sound. Since most of these softwares (such as Ircam's pm2, additive or AudioSculpt) output this representation in the form of an SDIF file, ircamdescriptor needs the SDIF loadsdif library in order to read the results.

The format of the SDIF file that are required are:

Sinusoidal analysis can be either harmonic, stored as 1HRM frame types and 1HRM matrix types, or inharmonic, stored as 1TRC frame types and 1TRC matrix types, both with the format: 1st column= Index, 2nd column = Frequency, 3rd = Amplitude, 4th = Phase.

Fundamental frequency must be stored with 1FQ0 frame types and 1FQ0 matrix types (1st column= F0 estimate)

It is important to understand that when using an external SDIF file, ircamdescriptor will synchronize itself with the SDIF frame rate (the same analysis time, i.e. the center of the analysis windows, will be used for the non-harmonic features). In order to have a correct processing, ircamdescriptor needs to know the window length that was used for the SDIF file. This window length is derived by ircamdescriptor by considering that the time tag of the first non-zero time frame in the SDIF file corresponds to the middle of the analysis window. The length of the window is therefore taken has twice this time.

About the output files

Ircamdescriptor computes a large set of audio features from the signal. The whole set of features is described into the document "A Large Set of Audio Features for Sound Description" available at the following URL: http://www.ircam.fr/anasynt/peeters/ARTICLES/Peeters_2003_cuidadoaudiofeatures.pdf.

Ircamdescriptor can output the computed feature value in three different formats (depending on your application one or the other may be preferable):

1. A text file with the temporal modeling values of each selected descriptor, for all dimensions and all temporal modeling segments.
2. An SDIF file containing both short-term and temporal modeling values for all descriptors. For the SDIF storage convention used by `ircamdescriptor`, see <http://sdif.sourceforge.net/descriptor-types>
3. A Matlab `.mat` file (you can load it in Matlab) containing the whole set of descriptor values over time, including short-term values and temporal modeling values.

Its format is a structure starting from `D` (Descriptor) with fields `DS` (Spectral Descriptors), `DH` (Harmonic), `DP` (Perceptual), `DN` (Noise Shape), `DERB` (Erb scale based), `DT` (Temporal), `DE` (Energy). The sub-fields of these families are the descriptors themselves. For example the Spectral Centroid is stored in `D.DS.i_sc_v`, and has the following subfields:

```

        name: 'SpectralCentroid'
        value: [47x6 double]
        dooutput: 1
        docompute: 1
        depend_on: ''
        type: 0
        nb_el: 6
        nb_dim: 1
        nb_var: 6
        temp_mod: [1x2 struct]
```

The meaning of the fields is the following:

<code>name:</code>	refers to the name of the audio features
<code>value:</code>	is a vector or a matrix containing the frame-wise feature values. Rows correspond to analysis frames and columns to the dimensions or variations of the feature (see below).
<code>docompute:</code>	indicates if the audio features have been computed (1) or not (0).
<code>depend_on:</code>	this is not to be used by the user
<code>type:</code>	is either 0 (for instantaneous values over time, such as Spectral Centroid) or -1 (for global values, such as Fluctuation Strength),
<code>nb_el:</code>	indicates the number of frame-wise descriptor elements (dimensions + variations).
<code>nb_dim:</code>	indicates the number of frame-wise descriptor dimensions. Dimensions correspond to either different frequency bands (such as in Spectral Flatness Measure) or to different coefficients (such as MFCCs or Autocorrelation).
<code>nb_var:</code>	indicates the number of frame-wise descriptor variations. Variations correspond to different definitions of the same descriptor (e.g. for the Spectral Centroid).
<code>temp_mod:</code>	contains an array of structures storing the temporal modeling data (see below).

A specific feature, 'TimeFrame' is given in all cases. This feature indicates the time of the frame on which the analysis has been performed. This can be useful for example to display the time evolution of a specific feature. For example, to display the time evolution of the first SpectralCentroid features, simply type under Matlab:

```
plot(D.DT.i_timeframe_v.value, D.DS.i_sc_v.value(:,1))
```

Descriptors can follow time scales other than the one defined by `D.DT.i_timeframe_v`. In that case, they will include an extra field `timeframe` in the above structure containing the time tags in seconds. This is the case for example for the Energy Envelope (`D.DE.i_enenv_v`).

In the case temporal modeling is performed (as defined by the configuration file `temporal_modeling.txt`, explained below), each element of the `temp_mod` subfield contains a structure with the following form:

```
name: 'LoudnessWeightedMean'
tm_length: 1
tm_hopsize: 0.5
timeframe: [0.5 1 1.5 2 2.5 3 3.5]
value: [7x6 double]
```

The meaning of the fields is the following:

<code>name:</code>	name of the temporal modeling.
<code>tm_length:</code>	length, in seconds, of the temporal modeling segments, assumed constant in the present version.
<code>tm_hopsize:</code>	hop size, in seconds, of the temporal modeling segments, assumed constant in the present version.
<code>timeframe:</code>	time tags at the center of each temporal modeling segment.
<code>value:</code>	value of the temporal modeling. Rows correspond to temporal modeling segments and columns to the dimensionality of the short-time feature.

About the ordering of the features in the output files

A feature can be a single element (a one-dimensional global feature), or a multi-dimensional matrix. In this case, the feature can take values over time (instantaneous features) or can be a multi-dimensional feature (at each time the feature is represented by a vector).

Examples of global/mono-dimensional features: `FluctuationStrength`, `Roughness`

Example of global/multi-dimensional features: `BandFluctuationStrength`, `BandRoughness`

Example of instantaneous/mono-dimensional features : `Noisiness`, `Inharmonicity`, `SignalZeroCrossingRate`

Example of instantaneous/multi-dimensional features: `MFCC`, `SpectralFlatness`, `SignalAutoCorrelation`.

In this case, each element of the matrix represents a specific coefficient (a specific MFCC coefficient for example).

A specific case concerns the audio features that are computed using several scales (Amplitude scale, Frequency scale. In this case, in order to reduce the number of features, all the variations (choice of the scale) are combined to compose a single multi-dimensional feature. For example the 6 versions of the SpectralCentroid (3 amplitude scales, 2 frequency scales) are stored as a single multi-dimensional feature with 6 dimensions which the following ordering

```
coefficient 1: lin-ampl/lin-freq,
coefficient 2: power-ampl/lin-freq,
coefficient 3: log-ampl/lin-freq,
coefficient 4: lin-ampl/log-freq,
coefficient 5: power-ampl/log-freq,
coefficient 6: log-ampl/log-freq
```

A specific case occurs when the feature is multi-dimensional by definition (such as the tristimulus which has 3 dimensions) and is computed using various scales (amplitude scales). In this case the ordering is the following:

coefficient 1 to 3: tristimulus in lin-amplitude
 coefficient 4 to 6: tristimulus in power-amplitude
 coefficient 7 to 9: tristimulus in log-amplitude

Features such as the 'HarmonicSpectralDeviation', 'OddToEvenHarmonicRatio' or the 'HarmonicSpectralVariation' do not depend on frequency. In this case, the 3 dimensions which are indicated correspond to variation of the amplitude scale:

coefficient 1: in lin-amplitude
 coefficient 2: in power-amplitude
 coefficient 3: in log-amplitude

Choice of the audio features to compute

The choice of the audio features to compute is done by means of a text file.

Running `ircamdescriptor -a writelist` will write a text file in the current folder, named `list_descriptors.txt`. This file contains on the first column the name of the features and a flag 0 or 1 on the second column. Setting this flag to 0 will force the program to bypass the computation of that feature. This can be useful if you want to save computation time (such as avoiding the time-consuming computation of the ERB bands). Note however that some features are mandatory because of dependency between features (such 'TimeFrame' and 'Loudness').

After editing and saving this list run `ircamdescriptor -a readlist`. This will generate the final configuration file `list_descriptors.mat` which contains all the parameters.

Then simply run `ircamdescriptor -a extraction -i filename` to compute the list of selected features. If you want to change the features that are computed do the previous process again: 1) edit the descriptor list text file, 2) save it and 3) run `ircamdescriptor -a readlist`.

Choice of the temporal modelings

The temporal modelings to be applied on the short-time features are controlled via the provided `temporal_modeling.txt` text file. A common configuration is shown here:

#	Name	Length	Hopsize	ApplyOn
#	-----			-----
	LoudnessWeightedMean	1	0.5	all
	LoudnessWeightedStandardDeviation	1	0.5	all
	ModulationAmplitude	f	0	EnergyEnvelope, FundamentalFrequency
	ModulationFrequency	f	0	EnergyEnvelope, FundamentalFrequency
	LogAttackTime	f	0	EnergyEnvelope
	TemporalIncrease	f	0	EnergyEnvelope
	TemporalDecrease	f	0	EnergyEnvelope
	TemporalCentroid	f	0	EnergyEnvelope
	EffectiveDuration	f	0	EnergyEnvelope

The first column indicates the name of the desired temporal modeling. The list above contains all temporal modelings implemented in the current version. The second column indicates the length, in seconds, of the temporal modeling segments (sometimes called texture windows). If an 'f' is entered here, the temporal modeling is performed on the whole input sound file. The third column indicates the hop size of the segments, in seconds. If the length is file-wise (f) the hop size is ignored. Otherwise, the hop size must be greater than zero. The fourth column contains a list,

separated by commas, of the descriptor names upon which each temporal modeling is to be applied. If `all` is indicated, the temporal modeling is applied to all descriptors.

Usage

First run

```
ircamdescriptor -a writelist
```

This will generate a text file with the list of all descriptors the program can compute. The first column is the descriptor name, the second one is either 0 or 1. 0 means that the descriptor will not be computed, 1 means the descriptor will be computed. You can edit this text file to select the set of descriptors you want to compute.

Then run

```
ircamdescriptor -a readlist
```

The program will then read the text file you have just edited, and configure itself for the first run.

For all the next runs (except if you want to select other descriptors), you run

```
ircamdescriptor -a extraction -i soundfile
```

When computing harmonic and/or noise shape features, you can specify the needed SDIF input files by

```
ircamdescriptor -a extraction -i soundfile -if0 f0file.sdif -iadd  
partialsfile.sdif
```

By default, SDIF output is not activated. To store the computed descriptors as an SDIF file, use

```
ircamdescriptor -a extraction -osdif outputfile.sdif
```